

T.M.

ON THREE

The Magazine For Apple III Owners and Users

Volume 2 - Number 1

\$3.00 U.S.



ON THREE Presents . . .

ONTIME Features:

- 1) Time & Date displayed whenever YOU want it!
- 2) Works with ALL programs!
- 3) Displays the Day, Month, Year Hour, Minute and Second.
- 4) Stop Watch/Timer Mode.
- 5) An unbelievable enhancement to your Apple /// system.

Calling all of you Time-Conscious professionals: Before ONTIME™ the only way you could find out the time and date from within some programs is if you looked on your watch!

ONTIME, the incredible new product from **ON THREE Magazine**, will conquer this problem by giving your Apple /// a great new feature: With the ONTIME clock driver installed you can display the Time & Date whenever YOU want it!

A simple keystroke combination will turn the clock display on, another will turn it off. One more turns on a stopwatch that can count down by tenths of a second! This is extremely useful in monitoring how long you've been on the telephone, or using a computer service like the Source.

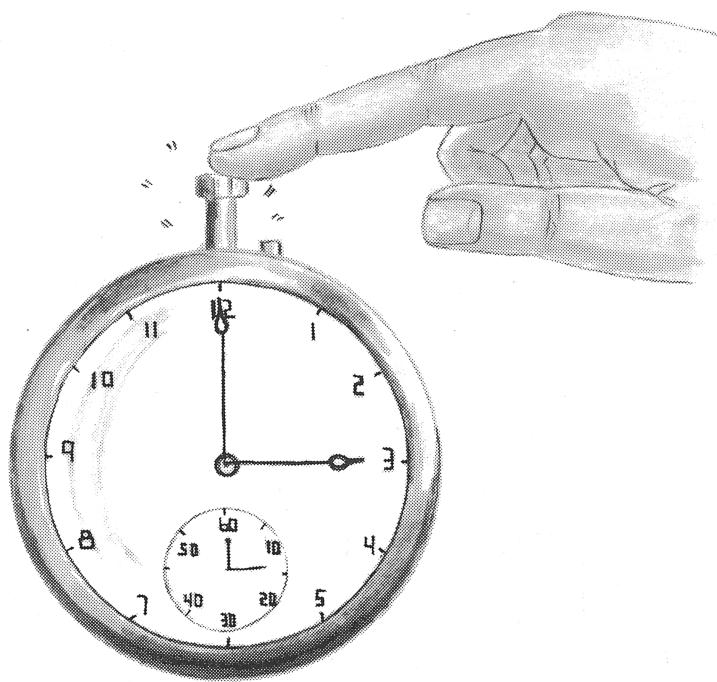
The best part about ONTIME is that it operates in the 'background' - continuously displaying the Time & Date while you are doing something else. Thus you can be working on your spreadsheet, typing a letter with your word processor or even printing out a financial report and ONTIME will continuously update the Time & Date information on the screen.

Completely compatible with all Apple /// programs that run under SOS, ONTIME is sold exclusively through **ON THREE Magazine** for only \$24.95. Please add \$1.50 for shipping and handling, California residents must also add 6% sales tax.

ONTIME requires an ON THREE O'Clock clock/ calendar or equivalent to work.

Special Deal! For a limited time you can purchase the ONTIME - ON THREE O'Clock combination for only \$59.90. That's five dollars off! If you've been waiting for a reason to get a clock — ONTIME is it!

ONTIME Clock Driver



ON THREE

/// TABLE OF CONTENTS ///

The Editor's Block:

Please Type Or Print Clearly
Name: _____
Company Name: _____
Address: _____
City: _____

Check One: _____ Check _____ Money Order _____ Visa _____ Master Card
Card #: _____ Exp. Date: _____

Ask THREE: 3 (Letters to the Editor) 4

Please send me the following items:	
Disk Of the Month #1	#2
ONTIME - ON THREE O'Clock	#3
ONTIME	#4
ON THREE O'Clock	#5 @ \$9.95 apiece + \$1.50 (SH)
Lazarus III Version 2.0	#59.90 total + \$3.00 (SH)
DRAW ON III	@ \$24.95 apiece + \$1.50 (SH)
(Lazarus & DRAW ON Require an Apple III plus or an Apple III with 256K)	@ \$39.95 apiece + \$2.50 (SH)
CROSSWORD-SCRAMBLER	@ \$29.95 apiece + \$1.50 (SH)
Micro-Sci Gameport	@ \$129.00 apiece + \$3.00 (SH)
Micro-Sci Disk Drives	A3 @ \$299 A73 @ \$409 A143 @ \$509
	Add \$6.50 for postage and handling for each drive ordered
	California Residents add 6% sales tax to total of above items
	Subscription (12 issues) to ON THREE @ \$30 = TOTAL =
Please remember to sign and enclose your check or credit card number.	
Allow 4 weeks for delivery.	
Have the subscription begin with	January Feb-March April-May June-July '83
Calc: 18
 25
 32
 42
 45

THREE Tips: 48 Dennis Cohen 48

See ads for: Micro-Sci disk drives, Gameport, Lazarus //, ONTIME, CROSSWORD-SCRAMBLER and the Disks Of the Month throughout the magazine.

Next Issue in ON THREE

Reviews on: Word Juggler, Apple Speller //, BPI... Family Finance... SOS Drivers & CP/M... ON The Phone ... WPL — From ALL angles ... Business Basic Program Lister ... Pieces of Lazarus? ... and more!

Volume 2 Number 1

Editor/Publisher:

Bob Consorti

Asst. to the Editor:

Suzanne M. Salazar

Interior Artwork:

Virginia Carol

Typesetting Services:

The Typesetting Company
Canoga Park, California

Printing Services:

Ojai Printing &
Publishing Company
Ojai, California

ON THREE - THE Reference Source for the Apple // is published bi-monthly by **ON THREE**, 5550 Telegraph Road Suite B-4, Ventura, California 93006.

For a copy of our Author Guidelines, please send a self-addressed stamped envelope (20 cents) to the above address.

Subscription information: U.S. - \$30 for 12 issues. For First Class Mail, please remit an extra \$10.

All foreign subscriptions should include additional postage in the following amounts:

\$8 for Canada and Mexico
\$12 for Central America - Caribbean
\$16 for South America - Europe - Africa
\$19 for Asia - Pacific Islands - Australia

Funds must be remitted in U.S. dollars drawn on a U.S. bank.

Group purchases must have one mailing address.

Return postage must accompany all manuscripts, drawings, and diskettes submitted if they are to be returned, and no responsibility can be assumed for unsolicited materials.

All letters sent to **ON THREE** will be treated as unconditionally assigned for publication and are subject to **ON THREE**'s right to edit and comment editorially.

Dealer inquiries are welcomed. Please write to the above address for volume pricing and terms.

ON THREE is a registered trademark of **ON THREE**. Apple, Apple II, Apple //, Apple // plus, Lisa, MacIntosh, Disk // and ProFile are all registered trademarks of Apple Computer, Inc. Micro-Sci, Gameport // are registered trademarks of Standum Controls, Inc. ON THREE O'Clock, Lazarus //, Draw ON // and ONTIME are registered trademarks of **ON THREE**. The Apple // Magazine. Opinions expressed in this magazine are those of the individual authors and not necessarily those of **ON THREE**. Entire contents Copyright © 1983, 1984 by **ON THREE**. All rights reserved.

ON THREE Presents . . .

ONTIME Features:

- 1) Time & Date displayed whenever YOU want it!
- 2) Works with ALL programs!
- 3) Displays the Day, Month, Year Minute and Second.
- 4) Stop Watch/Timer Mode.
- 5) An unbelievable enhancement Apple /// system.

Calling all of you Time-Conscious professionals: Before ONTIME™ the way you could find out the time and from within some programs is looked on your watch!

ONTIME, the incredible new program solves this problem by giving your Apple /// system the ability to display the Time and Date.

A simple keystroke combination will turn ONTIME on. One more turns on a stopwatch timer which is extremely useful in monitoring how long it takes to run a computer service like the Source.

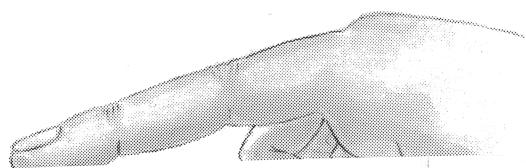
The best part about ONTIME is displaying the Time & Date while working on your spreadsheet, typing out a financial report and ONTIME will display the information on the screen.

Completely compatible with all Apple // systems, ONTIME is sold exclusively through **ON THREE Magazine** for only \$24.95. Please add \$1.50 for shipping and handling, California residents must also add 6% sales tax.

ONTIME requires an ON THREE O'Clock clock/ calendar or equivalent to work.

Special Deal! For a limited time you can purchase the ONTIME - ON THREE O'Clock combination for only \$59.90. That's five dollars off! If you've been waiting for a reason to get a clock — ONTIME is it!

ONTIME Clock Driver



ON THREE

/// TABLE OF CONTENTS ///

The Editor's Block: Bob Consorti	Ask THREE: 3 (Letters to the Editor)	4
---	---	---

/// FEATURES ///

Why Is Everyone So Excited About Lazarus?: Bob Consorti	So The Apple /// Is Only A Business Computer, Huh?: 12 David Cortopassi	18
---	---	----

PASCALCULATOR: Math On Demand In Apple /// Pascal. Al Evans	PFS - WPL Merge: 21 Jay Merritt	25
---	--	----

Apple /// - The User's Micro: Kevin FitzMaurice	Automating Access ///: 29 Ken Johnson	32
--	--	----

Slots Of Fun: Ben McCaw	35
----------------------------------	----

/// DEPARTMENTS ///

Would You Believe][to the Max?: Al Evans	Decision Support With VisiCalc: 36 Llona Cunningham	38
---	--	----

The Apple Writer WHAT?: Sharon Webb	New Products Received: 41 Bob Consorti	42
--	---	----

Book Review Dana E. Wilson, M.D.	Call THREE: Hot Line	45
--	----------------------------	----

THREE Tips:	Three Shorts — Fini! 48 Dennis Cohen	48
-------------------	---	----

See ads for: Micro-Sci disk drives, Gameport, Lazarus ///, ONTIME, CROSSWORD-SCRAMBLER and the Disks Of the Month throughout the magazine.

Next Issue in ON THREE

Reviews on: Word Juggler, Apple Speller ///, BPI . . . Family Finance . . . SOS Drivers & CP/M . . . ON The Phone . . . WPL — From ALL angles . . . Business Basic Program Lister . . . Pieces of Lazarus? . . . and more!

Volume 2 Number 1

Editor/Publisher:
Bob Consorti

Asst. to the Editor:
Suzanne M. Salazar

Interior Artwork:
Virginia Carol

Typesetting Services:
The Typesetting Company
Canoga Park, California

Printing Services:
Ojai Printing &
Publishing Company
Ojai, California

ON THREE - THE Reference Source for the Apple /// is published bi-monthly by **ON THREE**, 5550 Telegraph Road Suite B-4, Ventura, California 93006.

For a copy of our Author Guidelines, please send a self-addressed stamped envelope (20 cents) to the above address.

Subscription information: U.S. - \$30 for 12 issues. For First Class Mail, please remit an extra \$10.

All foreign subscriptions should include additional postage in the following amounts:

\$8 for Canada and Mexico
\$12 for Central America - Caribbean
\$16 for South America - Europe - Africa
\$19 for Asia - Pacific Islands - Australia

Funds must be remitted in U.S. dollars drawn on a U.S. bank.

Group purchases must have one mailing address.

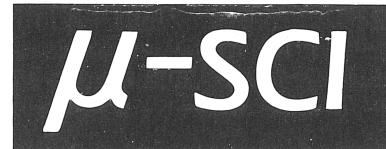
Return postage must accompany all manuscripts, drawings, and diskettes submitted if they are to be returned, and no responsibility can be assumed for unsolicited materials.

All letters sent to **ON THREE** will be treated as unconditionally assigned for publication and are subject to **ON THREE**'s right to edit and comment editorially.

Dealer inquiries are welcomed. Please write to the above address for volume pricing and terms.

ON THREE is a registered trademark of **ON THREE**. Apple, Apple][, Apple ///, Apple /// plus, Lisa, MacIntosh, Disk /// and ProFile are all registered trademarks of Apple Computer, Inc. Micro-Sci, Gameport /// are registered trademarks of Standum Controls, Inc. ON THREE O'Clock, Lazarus ///, Draw ON /// and ONTIME are registered trademarks of **ON THREE** - The Apple /// Magazine. Opinions expressed in this magazine are those of the individual authors and not necessarily those of **ON THREE**. Entire contents Copyright ©1983, 1984 by **ON THREE**. All rights reserved.

ON THREE Presents ...



MICRO-SCI

Micro-Sci Disk Drives

Every once in a while a product appears that is so good **ON THREE** decides to offer it for sale to our readers. **The Micro-Sci** line of disk drives (and the **Gameport //**) are the first of these superior type products. Byte for byte, these drives offer greater speed and more value than any comparable drive on the market today. If you are looking into purchasing an external disk drive for your **///**, **ON THREE** encourages you to look into this fantastic product line.

Expanding disk storage on the Apple **///** can be an expensive proposition.

But **Micro-Sci** has a better proposition for you, because our disk drives for the Apple **///** give you greater capacity and performance for every dollar spent.

And there are no compatibility problems. The A3 is a direct replacement for Disk **///** drives, and the 70-track A73 and 140-track A143 are supplied with a driver that is easily added to the SOS driver module, affording extra storage and fast seek rates for all of the programs that run under SOS.

Talk about compatible! All three are the same size as your built-in drive and they use the same diskettes!

Are all of your slots full? Don't worry, these drives plug right into the back of your **///** and they don't need a power cord! Up to three extra disk drives can be daisy-chained and they can be mixed in any combination of Disk **///**, A3, A73 or A143.

The A3 offers identical capacity to the Disk **///** and is an excellent choice for a second disk compatibility in the Apple **II** emulation mode.

At 286 KBytes, the A73 has double the capacity of the Disk **///** while the **A143 packs 572 KBytes** of data onto a diskette. With over **half a megabyte** of storage space, the A143 makes a truly viable backup device for the Profile Hard Disk.

With that large a capacity, many people find that they **don't need a hard disk!** Since up to three A143's can be used with your **///**, you can have over one and three quarter megabytes of data on-line at all times!

ON THREE is pleased to announce the following low, low prices on these great disk drives.

	A3	A73	A143
Suggested List Price:	\$379	\$529	\$659
ON THREE Price:	\$299	\$409	\$509
Savings	\$80	\$120	\$150

To order, use the attached envelope and add \$6.50 for postage and handling for each drive ordered. Please allow four weeks for delivery.

Gameport //

You don't have to be chained to your job, and neither does your Apple **///**. After the working day is done, release your computer into the exhilarating world of adventure and challenge with a **Gameport //** from **Micro-Sci**. The new Gameport **//** game controller adapter lets you use game paddles, joysticks and all your favorite Apple **II** amusement packages with your Apple **///** computer. The Gameport **//** is easy to use and simple to install - your only challenge is to conquer the invaders!

The Gameport //

- Allows all games written for the Apple **II** to be used on the Apple **///**.
- Works with all Apple **II** game paddles and joysticks.
- Allows programs which require a game I/O protection key to run in Apple **II** emulation mode.
- Can be installed in any slot.
- Does not interfere with the normal operation of the Apple **///**.
- Package includes: Gameport **//** board, Apple **II** Emulation Modification Diskette and complete, easy-to-follow instructions (Apple **II** game controllers not included).

ON THREE proudly sells the **Gameport //** by **Micro-Sci**. For only \$59.95 you can now get the best that the Apple **///** and the Apple **II** has to offer. That's \$15 off the suggested list price so don't be left out, place your order today! Please use the attached envelope for ordering and remember to add \$2.50 for postage and handling.

The Editors Block:

Bob Consorti

So much news about the Apple /// and so little space to tell you in! The biggest announcement is the long awaited Apple /// plus. The big question is what's different? Not much, the only real reason for the /// plus is because the Apple /// never met FCC regulations (Ever try to watch TV in the same room with an Apple ///?).

The /// plus sports an improved back panel design that helps reduce EMI (Electro-Magnetic Interference) by providing connectors that will hold interface cabling together and not fall apart at the slightest movement. Additional shielding throughout the chassis lets the /// plus meet the FCC regs. A clock is included in the /// plus, as this is a very handy feature.

The major changes are cosmetic. There is a new keyboard design, and it's very similar to the //e's. Accountants don't despair, they left on the numeric keypad! Oops, I almost forgot - the nameplate now reads "/// plus".

The last change is very interesting but it doesn't quite meet up to the hype surrounding it. There is a new interlace mode that will double the vertical resolution of the graphics display from 560 by 192 pixels, to 560 by 384 pixels. For you number buffs out there, that's a greater resolution than the new Macintosh.

The only problems with this are: 1) Apple has yet to supply a monitor that can display that resolution, and 2) There is no software available that uses the new mode. **Problem #2** will be solved in time, as soon as Apple releases the necessary routines I'm sure that developers will write packages supporting the new graphics mode. **Problem #1** is a very difficult one to solve.

To display a screen of that resolution, an extremely slow phosphor monitor is needed. Without a slow speed phosphor, the image on the screen flickers very noticeably. Distracting enough to cause eyestrain, it clearly shows that the current Monitor /// isn't any good for this graphics mode. Since my contacts tell me that Apple isn't going to come out with one - they have destroyed any potential market for programs that use the new graphics mode. It's like coming out with a 100 megabyte drive for the /// but not making an interface card.

So why did Apple give the /// plus a feature which has no use? Well, it does have one use. It not only doubles the vertical resolution of the graphics screen, it "Interlaces" the image of the text screen. This makes for clearly readable text. If you were ever disappointed about the "Grainy" looking display of the ///, this new item will cure that. With interlace on, the text on the screen is of very high quality.

All of the "features" the /// plus offers can be purchased for your present Apple /// at a reasonable cost. Your dealer can give you more information about this. Upgrading (new keyboard, interlace) isn't really necessary as the machines are completely compatible. Everything that will run on a /// will run on a /// plus and vice-versa.

What the /// plus doesn't have is 512K of RAM, expanded floppy disks and a clear message from Apple that they are going to support the new machine. You are probably aware that Apple has come out with a new generation of personal computer - The **Macintosh**. Priced directly in the ///'s market (it's actually less

expensive), they have a coordinated marketing strategy for this product.

They have already begun a nationwide television advertisement campaign for the Mac and more is on the way! Millions of dollars have been earmarked for promoting the Mac in every possible way. The media hype for this computer has been tremendous. True, it is a very nice little machine, but so is the /// - and where is the support for it?

It's been said that the Mac is the machine that will compete head to head with the IBM PC so I can understand why Apple is promoting it so much. On the other hand, people have been saying that if Apple supported the /// from the beginning - the IBM PC would have flopped. Why must Apple square off only the Mac against the PC? Doesn't it make more sense to nationally promote the //e and /// in addition to the Mac? Wouldn't it be better if Apple used a number of machines to fight off the PC?

I have my opinions and I'm sure that you have yours. It seems that petty rivalries within Apple have caused the ///'s problems and I don't see an end to it. Every paper and journal I read says the /// is a dud. Almost everyone I meet has never heard of the Apple /// yet they know that there is an Apple // and an IBM PC. It's easy to call a product a dud when you know nothing about it. If the /// is ever promoted as the marvelous tool that it is, it will no doubt sell very well.

I'm not really harping on the people in the Apple /// group inside Apple, they have to follow policies of the company and if no resources are allocated for the /// they can't do much. It's almost funny! For the resources that they ARE given they have done a great job. The people in the Apple /// group are among the brightest in Apple - they have to be, to market a product that is getting little company support.

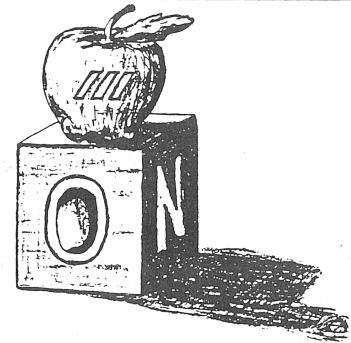
I hope that these paragraphs above will stimulate some thinking on the Apple ///'s problems. It's not my job to create controversy, the /// already has that. I'm simply trying to help the /// assume a respectable place in the marketplace. If you have any comments we will gladly publish them, so please - let's talk about the ///.

The Real Good Stuff...

Rumors for the /// include a mouse in the near future, a 512K upgrade before the end of the year, and a megabyte of static RAM (it doesn't lose its contents when the computer is turned off) as an ultra fast disk drive. The mouse is a certainty and should be announced very soon. The memory expansion and RAM disk will probably have to wait for the new 256K memory chips. This may not occur until the end of the year.

GREAT news for everyone with a 128K Apple ///. On January 15 of this year, Apple cut the price on the 256K memory upgrade from \$600 to only \$300!! One of the major reasons they did this was to encourage everyone to upgrade to a 256K configuration. Since many programs for the /// need a 256K machine it really is a necessary option. Our **Lazarus /// Version 2.0** and our new **Draw ON ///** both require a 256K machine, these new

Editor's Block continued on page 11.



Ask THREE: (Letters to the Editor)

Dear ON THREE,

We are a Law Firm engaged in an area of practice whereby we fill out numerous United States Government forms. As you may guess, we use a wide variety of Government forms that, for example, may ask for a person's date of birth on each and everyone; but ask for the date to be placed in a different position on each of the various forms. Is there a data base or word processing software program available that will readily allow us to print from a single reference or data source for each client to any one or all of the various forms that may be required? Apple Writer ///'s word processing language seems to have come close; but we have not perfected it to where it is reliable as of yet. I guess I am looking for an easier way out.

Thank you very much for your consideration, I have greatly enjoyed your magazine thus far. Keep up the good work.

Sincerely,

William H. Hoover
Texas

Dear Mr. Hoover:

With regard to your question, there is an excellent new data base program called KEYSTROKE. The KEYSTROKE Data Base and the KEYSTROKE Report Generator combination would probably handle the needs that you expressed.

We are also going to publish a number of articles on just how to do that sort of thing using WPL. If you don't want to spend a few hundred dollars, this is the inexpensive way out.

Dear Mr. Consorti:

Why does Apple Writer version 1.0 (and 2.0 - Ed.) only allow 12 lines of input while the rest of the screen is blank? Why can't I have 22 or 24 lines of input?

Another question is why do I get a volume not found error when I try to invoke "BGRAF.INV". I have the SOS.KERNEL, SOS.DRIVER, SOS.INTERP, and BGRAF.INV on my disk. I also have the following drivers, .CONSOLE, .AUDIO, .GRAFIX, and .PRINTER. With my graphics program I have the first lines 10 and 20 as follows:

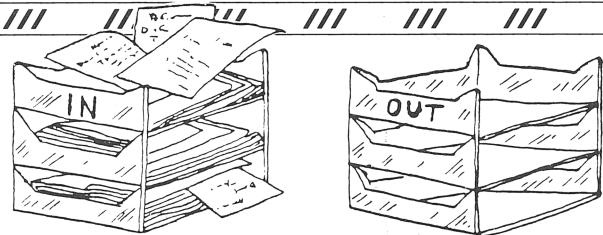
```
10 OPEN #1,"GRAFIX"
20 INVOKE"BGRAF.INV"
```

Then when I run the program I get a volume not found error in 20.

Only when I change line 20 to read: 20 INVOKE ".D1/BGRAF.INV." will my program run. What gives?

Sincerely yours,

Dale Means
Indiana



Dear Mr. Means:

Apple Writer /// likes to keep the cursor centered in the middle of the screen, and that is why there are only twelve lines of input text available while the rest of the screen is blank. The program is written that way so there is no real way to change it except through letters like this.

In reference to your question about the Basic program, when you try to invoke the BGRAF invokable module, your program gives you the "volume not found error". This is due to the fact that you probably have the prefix set to .D2 or some other pathname in your system.

When you change line 20 to tell it to look on .D1 for the invokable, it finds it.

If you did not wish to put the .D1 in line 20, you will have to first set the prefix to be equal to .D1. For example, line 5 would read: PREFIX\$= ".D1". Then, when it comes time for line 20 to run, the program would invoke BGRAF.INV from the disk in the currently selected prefix drive which is .D1.

Gentlemen:

Last year I purchased a Micro-Sci A143 through you, and it works like a charm—first time, every time, in contrast to some of the hardware by other manufacturers I have encountered. I also have the clock you sell and am delighted with it. It is to be hoped that all your devices will be this reliable and economical to purchase.

Your software reviews are also appreciated—both the formal articles and the readers' comments in the letters column, so keep up the fine work! How about a review on the bewildering variety of printers now available, with special reference to price and utility for the ///? A number of manufacturers are now producing hard disk drives (e.g. Davong) adaptable to the /// at a fraction of Apple's price. Any comments or experience on these will be most interesting.

Sincerely,

David H. Ballard
Michigan

Dear Mr. Ballard:

A number of hardware reviews on printers and a special on hard disks will be coming up in the near future. We have our favorites and a couple of bombs that we will let everyone know about.

Thank you for your thoughtful comments and for your continued support of ON THREE. It is greatly appreciated.

Dear Bob:

What a fantastic magazine and service for Apple /// users you have! As a new subscriber, I have just finished reading the

April/May issue and continue to be delighted with the overall quality of your (our!) magazine. I can't help but echo the voices of the other Apple /// users who have written in expressing how refreshing it is to finally find a magazine that is dedicated to the ///. Keep up the good work!

I have had my system now for about four months and although I consider myself to be a novice, I do have a couple of things to pass on to you and your readers that may be of some interest.

Did you know that the built in diagnostic tests can be initiated from the keyboard and run continuously? Here's what you do:

STEP 1: While holding down the CONTROL OPEN APPLE keys press and release the RESET key.

The screen should now have "→" appearing in the upper left corner of the screen with the underline blinking.

STEP 2: Type F6E6G then RETURN.

Your system should now be continuously executing the built in test and alternately displaying a series of test patterns on the screen.

Unfortunately, I do not know what all these patterns mean except that if any character besides a dot appears in the matrix of dots entitled DIAGNOSTIC RAM, one or more of the RAM chips is bad. The location of the "non-dot" character also defines the location of the bad chip.

Did you know that we can also program in Applesoft and Integer Basic? You are probably wondering why anyone might want to do this on the /// but my interest stems from the fact that my eleven year old daughter sometimes wants to try out Apple // programs that have appeared in a computer magazine that she gets and I didn't want to have to translate them into Business Basic. Anyway here's what you do:

STEP 1: Boot the emulation diskette and select the desire language from the configuration menu.

STEP 2: Press RETURN, and while the drive is whirring, press the RESET key.

You should now have the appropriate prompt for the language you selected (e.g."J" for Applesoft Basic and ">" for Integer Basic).

You are now in the programming mode and can enter Apple // Basic statements. At this point, however, I recommend that before you begin entering statements, you take a blank disk and format it in Apple // format so that any program that is developed can then be saved To do this simply insert a blank diskette into .D1 after STEP 2 and type INIT HELLO and then RETURN. The disk drive should whir as the disk is being formatted. After the disk drive stops, you should have a blank diskette that has been formatted in the Apple // format and onto which you can now save Apple // programs (and data). Also, after you have this disk with HELLO on it, further programming in Applesoft (or Integer) Basic can be accommodated without going thru STEP 2 above by simply inserting the disk after STEP 1 and pressing RETURN (i.e. by the normal Apple // program boot process).

I have to add here, however, that although the above technique has worked for me there may be some hidden flaw(s) in the above process as I am not familiar with the interworkings of the emulation mode. If you or one of your readers know of any or of a better approach to doing this I would like to hear about it.

Now for a few questions. I have obtained a copy of SOS 1.3 from my local dealer and have updated SOS.KERNEL and the drivers associated with my applications programs. However, is it also necessary to update any of the system related files such as SOS.INTERP?

Do you know of any way to convert an SOS formatted diskette into an Apple // format and vice versa? What I would like to be able to do is to read a DIF file that I created in VISICALC /// (under SOS) with an Apple // VISIPLOT/VISITREND program, create some plots, save them as a binary file, and then make a hard copy using Business Basic. I know that this sounds like along way around the barn but my printer (which is a GEMINI 10) is not supported by VISIPLOT. Do you know of any way that I can accomplish this?

Do you plan to review Apple Business Graphics /// in the near future? Because of the problems I'm having going back and forth between the native mode and the emulation mode that I just discussed, I'm considering purchasing this package. Any suggestions?

I'm also having problems getting my printer to respond properly to print control characters that I send it from Apple Writer ///, in particular when I try to switch between 132 characters per line back to either 96 or 80. At the beginning of my file I type CONTROL V, the appropriate control characters (which for the GEMINI 10 is ESCAPE B 1 for 80 characters per line), another CONTROL V, and then RETURN. For some reason, however, this does not seem to work and the printer continues to print in the 132 characters per line mode. My printer is connected via a UPIC. Any ideas?

As this letter has become too long, I had better "put a lid" on it. Congratulations again on your fine magazine. I anxiously await your next issue.

Sincerely,

Ken Jutzi
California

Dear Mr. Jutzi:

Thank you for your letter. You will not need to change SOS.INTERP, only SOS.KERNEL and the drivers. You can transfer files from your Apple /// disk to an Apple // format disk with an excellent utility called "SOSTRAN" from Sun Data Inc., 95 W. 100 South, Logan, Utah 84321, telephone number (801) 752-7631, for \$60.00.

The problems you are having regarding getting your printer to respond properly with certain control commands is not unique, and is, in fact, quite common. Apple Writer /// intercepts certain control characters before they are sent to the printer, and thus, they are not sent. There is no way around it with the normal printer driver that Apple supplies. However, Sun Data also has a package called "Printer Driver" which gets around the problems and enables you to use the full features of your printer.

Thank you again for your letter and for your support of ON THREE.

Dear Mr. Consorti:

Enclosed is a check for a one year subscription to ON THREE.

Finally a magazine devoted to the Apple ///! A few months ago, I would jump if I saw so much as a picture of an Apple /// in the computer magazines I subscribed to. A few articles on the Apple /// were out of the question back then, or so it obviously seemed to the publishers of computer magazines. But a Magazine on the Apple ///?! Such a thing would only be in a

/// /// /// /// /// /// /// /// /// /// /// /// /// /// ///

dream after eating a pizza.

Is it too late to purchase a Micro-Sci disk drive at a reduced price from you?

What is meant by fixed-media regarding the ProFile? Does it mean that files cannot be copied to a disk? Also, what are "interrupts"?

I read a letter in the April-May issue of your magazine by Mr. John Lomartire. He said that the .PRINTER driver did not work with NEC 7720. He said that the .RS232 driver had to be used to correct this. Why? This is how I set mine: 08 26 00 00 00 80 13 11 DF DF 80, and it works perfectly. I have changed the name of the driver to .NEC to avoid confusing it with other printers. I hope this will help Mr. Lomartire.

Though this would not be of much help, it might interest you. Typing PR#7 (I think it is the "7", but my emulation disk isn't working.) will cause the printer to type every letter you type in!

Is there any way of making the NEC 7720 capable of proportional spacing?

Could you publish a series of articles on assembly language? I noticed that you have articles on Basic and Pascal; I am somewhat "fluent" in these languages, therefore, I would like to learn assembly language. Also, I would like to know how "Assembling on the ///" works, especially the one in the April-May issue.

I have a book on assembly language that assumes everything, and has taught me nothing. Do you know of any books on assembly language that can be understood?

Yours truly,

Frank Hammers
Louisiana

Dear Mr. Hammers:

I enjoyed your letter very much, thank you for your enthusiastic comments.

We are still selling Micro-Sci drives at the reduced price, and you can order them either by telephoning directly with a Visa or MasterCard or by sending a regular order in.

With regard to your other question, "fixed-media" means that the disk cannot be removed from the drive as floppy disks can be taken out of the disk drive and then you can put a different floppy disk in. Files can be copied from the Profile or any other hard disk. There is complete compatibility within the varying disk drives.

An "interrupt" is something that stops the currently operating process of the computer so that it can do something else. For example, when you press a key on the Apple ///, the currently operating program is halted (temporarily), and the control of the computer is passed to something called the "interrupt handler" within the console driver. This then puts the value of the key press into the type-ahead buffer. When done, it returns control to the previously operating program. This happens so quickly that the user never knows that his program was stopped momentarily. Then, whenever the program asks the console driver for a character, instead of stopping the entire system and waiting for the person to press a key, the character comes from the typeahead buffer. Thus, the computer doesn't have to stop all the time to do things. You can type a whole line of commands at once and go have a cup of coffee while the computer works on those com-

mands that you typed in.

Another example of "interrupts" is whenever you print something to your printer. Since your Apple /// can send information faster than the printer can print it, it would be a waste of time to tie-up the entire Apple /// while the computer was printing.

When printing a large document, the Apple /// solves the problem by not sending the document straight to the printer, but, first, to a buffer inside the computer. When the printer is ready to accept another line of information from the document, it interrupts the computer. Now control is passed into the interrupt handler of the printer driver. This routine gets a line from the buffer and sends it to the printer. It then returns control to the previously operating program, which could have been, say, VisiCalc. Thus, you could have sent your spreadsheet to be printed, and it would then go into the buffer; and while you were doing something else, it would print out. This is called "concurrency", and is the only implementation on a micro-computer that I know of. The printer runs in the background while the computer can do something else in the meantime.

Thanks for the notes on the printer driver problems. We aren't familiar with the NEC7720, but we will ask around for you. A series on Assembly Language is coming up. This should help you in your quest for knowledge in Assembly language.

Additionally, Rodney Zak's book, "Programming the 6502" by Sybex Publishing is a good book to start off with for learning Assembly language programming.

Thank you again for your letter and for your support of ON THREE.

Dear Bob:

I just received Issue #4 and am impelled to share some of my experiences with ProFile. I was struck by the glaring difference between your reader's horror stories and the "press" given it through Apple dealers. It sounds as if there have been substantial problems with ProFile, and that its reliability has been overstated. On the other hand, the solution to these problems may be quite simple.

I have had three or four major ProFile head crashes and lost some non-essential data as a consequence of my own failure to keep absolutely current back-ups. I grew to have a familiar unpleasant sinking feeling in the pit of my stomach every time "damaged directory" appeared, which it did all too often.

My local dealer was supportive throughout, and never failed to come through. Apple relayed, through my dealer's repairman, that I had the worst ProFile, in the whole world! On crash #2, my Apple salesperson, since bumped "upstairs", gave me his own ProFile, which contained a number of goodies, including his own tax return! Being a gentleman I purged it without further examination.

Finally, Apple replaced the entire unit. At the same time I bought a surge suppressor. Six months later, I have had no further problems. We have had floods, earthquakes and electrical storms. I have no idea whether it was the new hard disk or the surge suppressor that solved the problem. On the positive side, I can recommend a hard disk crash as a good way to get "housecleaning" tasks done since it reminds one to discard unwanted files.

A few suggestions for further applications of Apple /// in your operations... Why not set up an ON THREE bulletin board? I

doubt that it would detract from the Correspondence section; doing otherwise ignores the communications capabilities of the Apple ///. Interesting exchanges could be saved and printed in **ON THREE**. Do you use the typesetting capabilities of the ///.

Apple Speller /// should be given a crack at the proofs of the magazine before they are "put to bed". Quite a few typographical errors have managed to slip through in the last issue.

Finally, you might consider an article on Catalyst structure and problems. It has been a delight to use, but I have had to do some barely-satisfactory patching and still have lingering problems with my parallel driver for the NEC 8023 which neither my dealer nor Quark has been able to solve.

ON THREE is already a tremendous success and it's getting better. Apple /// users like myself are reassured by the grassroots support that it represents. It is exciting to see the creativity that can be brought to bear on problems which have vexed all of us, and even more exciting to see problems defined for applications which I haven't even thought of! My warmest wishes for your continuing success!

Sincerely,

Dana E. Wilson, M.D.
Utah

Dear Mr. Wilson:

Thank you for your thoughts on the ProFile hard disk. As you probably know by now, we had similar problems that were also corrected with the addition of a surge suppressor. It seems a very valuable addition to any office with sensitive electronic equipment.

The **ON THREE** bulletin board will be set up next month and announced in the next issue. I also think some fairly interesting exchanges will occur on the system, and they will be included in future issues of **ON THREE**.

Our typesetting capabilities are such that once we get an article to be typeset, we add the typesetting commands, such as enhanced italics, and we then use a modem to transmit it directly to the typesetting machine.

Sorry about the last issue. I know a lot of typos did creep in. We didn't use our automated typesetting capabilities in the last issue as we wanted to test out a local typesetter and see a comparison - quality and such. He didn't measure up to our standards and we will not be using his service again.

We have an upcoming article on the use of Catalyst and overcoming some of its problems. Thank you again for your support of **ON THREE**.

Letter to:
Software Publishing Corp.

I enjoyed the latest issue of PFS: NEWS, particularly the information on PFS: WRITE and PFS: Solutions; but was disappointed that you are not advertising these as being available for the Apple ///. I have found PFS and PFS: REPORT to be very valuable additions to my Apple /// software library and of all the software I have, the PFS: Series is the easiest and quickest to use.

After hearing that PFS: WRITE was to be released, I told my Apple dealer that it will be one of the best word processors around and that he definitely should stock it. This, of course, was based only on my experience with the other PFS: Software.

Please do not stop providing software for the Apple ///. I

know you have to market where the "numbers" are, but don't forget where the "quality" is, the Apple ///.

Sincerely,

Howard L. Olien
Minnesota

P.S. I couldn't help but notice the Apple /// in the picture on page 4 of PFS: NEWS - "SPC: Quality People Making Quality Products."

Dear Mr. Olien:

Thank you for the copy of your letter to Software Publishing Corporation regarding the PFS series. I'm sure our other readers will like it.

Dear Mr. Consorti:

Most Apple /// owners are aware that their machines have not received widespread acceptance. This sad fact is being helped along by recent publications.

As you probably know, CONSUMER REPORTS recently published a fairly lengthy review of the most prominent currently available microcomputers, completely omitting the Apple ///. Such reviews could play a major part in the future on the ///; the reviewers' perception of the importance of a particular machine might be influenced by a sufficient public response. I have written a letter to the magazine (copy below) pointing out this omission; other readers of your magazine may wish to follow suit.

The first few issues of your publication have contained a number of items of interest to me and I am sure it will be a valuable resource to all Apple /// users. Keep up the good work!

Sincerely,

David L. Yerzley, M.D.
California

Letter to CONSUMER REPORTS:

Dear Editors:

I have read the section on personal computers in the September 1983 issue of Consumer Reports and found it thoughtful and well researched, in keeping with the standards which have resulted in my habitual consulting of your opinions before making most significant purchases.

One computer absent from your discussion which is, I think worth including is the Apple ///. Although it has not achieved the market acceptance of the IBM PC, of which it is a direct competitor, it has a number of strengths:

1) An extensive library of software which will run on the Apple /// is available.

In addition to its native operating system, SOS, an Apple // emulation program is supplied with the computer, allowing it to run most software written for the Apple //, //+ or //e using DOS 3.3. An optional circuit card allows the use of CP/M-based programs as well and includes Microsoft BASIC.

2) The uniquely elegant operating system of the Apple /// eliminates the need to keep track of the machine's physical configuration when manipulating files and performing other routine operations.

As the system is entirely RAM based, any existing machine can be updated to the most recent version simply by changing the system files on boot discs. (An update which became available after I had bought my machine was supplied free in the form of a program which automatically revised all boot discs.)

3) The Apple //, with 256K RAM, costs little more than an Apple //e with a few extra circuit cards and two disc drives but is considerably more powerful. It is significantly less expensive than the IBM PC yet its capabilities compare favorably with those of the PC.

The IBM PC and Apple // use different microprocessors; the 16 bit Intel 8088 used in the PC is touted by IBM as an "advanced feature" for personal computers. Most benchmark routines, however, require essentially the same time to run on either machine, and the 8088 handles input and output in 8 bit increments, as do the PC's peripheral devices. Although at present only a 256K memory is offered by the manufacturer, the Apple // 's architecture is capable of addressing up to 512K of RAM. Some features such as color capability, optional in the PC, and an RS-232-C serial interface, allowing direct connection with a modem, are standard in the Apple //.

4) Versions of BASIC and Pascal written specifically for the Apple // are sufficiently versatile to support custom programming of the machine for almost any imaginable business, scientific or engineering application.

5) Apple Computer has a well deserved reputation for the excellence of its customer service.

I experienced some hardware problems with my machine initially which were intermittent in nature and impossible to demonstrate to my dealer. Based entirely on my description of the problem, the dealer and Apple's area representatives cooperated in replacing a major portion of my system to eliminate the fault. In my prior years of experience as an aerospace engineer I found such acceptance of responsibility by an equipment supplier to be all too rare. (In my case several major rebuilds of the machine were required).

I suggest that such considerations make the Apple // an attractive choice for those who seek a powerful and versatile personal computer.

Sincerely,

David L. Yerzley, M.D.

Dear Dr. Yerzley:

Thank you for your letter regarding the CONSUMER REPORT's review of the currently available microcomputers. It is quite interesting and definitely thought provoking.

We thank you for your candor and support of ON THREE.

Dear Mr. Consorti:

First I would like to express my appreciation for ON THREE. You have done a fantastic job for a start up special interest magazine. Each issue is eagerly awaited. Thank you.

Assistance with three questions please! My son is studying Fortran in college. Is there a way to run Fortran on our // so he can use it in his studies? Secondly, is it possible to change the // 's keyboard from Sholes (QWERTY) to DVORAK? Last, would you consider a classified section for ON THREE so subscribers could offer for sale items peculiar to the //? It would be of great value for many users.

Keep up the good work.

Sincerely,

Richard A. Coomes
Kentucky

Dear Mr. Coomes:

Thank you for your letter. Regarding the Fortran problem, I have been assured by various personnel at Apple that a Fortran package for the Apple // will be forthcoming. The exact date I am not sure of yet. But, we will publish notice of it as soon as we hear.

Yes, it is possible to change the Apple // 's keyboard from Sholes to Dvorak. All you need to do is use the System Configuration Program and go into the Change System Parameters Menu and load in the keyboard layout file that is on the Utilities.Data disk called /UTILITIES/KEYBOARD.LAYOUT/DVORAK.

Lastly, you will need to gently pull each key cap off of the keyboard that is different and replace it in its proper Dvorak position.

I believe that Apple has a Dvorak keyboard available, but I am not sure of this.

We have considered a classified section in ON THREE, and it will more than likely be in future issue.

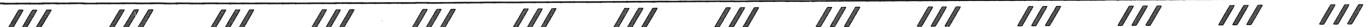
Thank you again for your support. I look forward to hearing from you in the future.

Dear Mr. Nichols:

Of all the articles in ON THREE, I enjoy yours the most. Each utility is VERY well documented and the Documentation and Test programs that are provided are really good. I especially enjoy the way each utility is demonstrated in both Basic and Pascal...this in itself provides me with a great working tutorial of the similarities and differences between the two languages.

You asked in the June-July issue what features I would like to see in the column. I for one would like to see the column expanded to include tutorials for assembly programming of the //. There are many good books now on the market for assembly programming of the 6502 microprocessor that is in the Apple //, the tutorial could take off where these books end and describe the enhanced indirect addressing capabilities of the //. I realize that much of this information is in the SOS Reference Manual but as you are aware the Manual is far from a tutorial.

A utility that I would like to see would be a File Type Converter. In other words I would like to be able to change file types, from say a data file to a font file. Is this possible outside of the Pascal environment? Also on Disk Of the Month #1 there is a whole set of Character Fonts. I would like to be able to use these in Apple Writer // (using the change character font command), but Apple Writer requires these files to be pascal data files. The PKASO card with an Epson allows me to print using any system font so it would really be nice if I could use those additional fonts. Also any idea how Catalyst works??? If it can run as a master interop controller lurking behind each of the programs on its menu, is there any reason why an automatic Apple // can't be set up. Such a master program would not only control the interops and loading of programs on the menu, but would tie the sequence and control of each program to the system clock so



products from Apple for the ///. I thought I would share what they sent with your readers in the event some of them did not receive the same brochure. Listed they are:

—Apple Serial Card /// to add an additional serial port to drive "nearly any standard serial device, such as modems, printers, plotters, etc.";

—Apple Writer /// (Version 2.0), and enhancement offering "more simplicity and new capabilities, including direct interface to Apple Speller ///";

—Apple BPI /// Business Accounting "...that takes advantage of the ProFile hard disk...includes General Accounting, Accounts Receivable, Accounts Payable, and Payroll.";

—Catalyst, Version 2.0, from Quark, "to put any Apple /// program on the ProFile hard disk and switch from one program to another by simply selecting the program from a simple menu."

What's more, Apple announced the formation of "an independent business unit dedicated solely to the task of supporting the Apple /// product line."

Again, thanks for a very good magazine. As a former resident and prosecutor in Ventura County (3 years ago) I was surprised to find my source of answers coming from familiar turf.

Looking forward to the next issue.

Sincerely,

Jack D. Wood
California

Dear Mr. Wood:

Thank you for the information on the new software listing for the ///. I'm sure all of our other readers would like to hear about it.

All of our Disks Of the Month after Disk Of the Month No. 1 included the Basic system on it. This means that you don't have to buy the Basic language system for the /// to run the graphics program in the magazine. Simply buy the DOM!

Thank you again for your support.

Dear ON THREE,

How can I find out more about "Patching Apple /// Pascal" by Dr. J. Jeppson? I would appreciate an address or any other help you can pass on.

Thank you,

Ron Joyce
Minnesota

Dear Mr. Joyce:

With regard to your question about "Patching Apple /// Pascal" by Dr. J. Jeppson, that was an article published in SOFTALK magazine in the February 1983 issue, page 190 through 198. You should be able to find a back issue at a local dealer.

Additionally, this is only valid for Apple /// Pascal version 1.0. Versions 1.1 and version 2.0 (2.0 is not currently available to end users) both have provisions for doing the same type of thing that Dr. Jeppson outlined in his article, and it is included with the Pascal system as a program called "PMOVE".

Dear Mr. Consorti:

Is there a Logo software program available for Apple ///?

Sincerely,

W. Warren
Kentucky

Dear Mr. Warren:

To the best of our knowledge, there is not a logo software program available for the Apple /// as yet. However, as soon as we hear of one we will pass on the pertinent information.

Dear Mr. Consorti,

As a recent subscriber to ON THREE, I was delighted with the amount of information you shared with your readers. At last there is a source where Apple /// users can go to when in need.

I have a few questions concerning the Apple ///.

1. Can the ON THREE O'Clock be used in Apple // emulation mode? If so, how would this be done?

2. Do you know of any packages that enable graphics dumps to printers? I have an NEC PC8023 Dot Matrix printer running through the UPIC card.

3. Can you recommend any average priced modems that could be used on the ///? I don't know much about modems and would appreciate an evaluations on some currently available modems.

Thank you for your time, and I look forward to receiving the next issue of ON THREE. You're doing a great job there, so keep up the good work!

Sincerely,

Ali Arjomand
California

Dear Mr. Arjomand:

Thank you for your letter your enthusiastic comments about ON THREE. They were very much appreciated.

Concerning your questions about the ON THREE O'Clock, it cannot be used in emulation mode due to the restrictions that emulation mode places on the C000 I/O expansion space.

To the best of my knowledge, there are no packages that will enable graphics dumps on your NEC PC8023 running through the UPIC card. I would recommend dumping the UPIC and getting a PKASO which has full text and graphics dump capabilities built in. We recommend the Smart-Modem by HAYES as the best modem on the market. There are cheaper ones on the market, and those companies are not going to be around. HAYES has been around. They're a proven leader in the field.

Thank you again for your letter. And thanks for your support of ON THREE.

Dear Bob:

I read with interest Mr. Nichols' article in the June/July issue

on formatting disks from Basic and Pascal. I cannot seem to get the appropriate files onto my Business Basic Diskette. I have the DOM and have transferred FORMAT.TEXT from side two. However, since FORMAT.TEXT is type PASTXT I can't do anything from Basic (type mismatch error). What have I done wrong?

Sincerely,

James P. Santelli
Minnesota

Dear Mr. Santelli:

The files you will need to format disks from within Basic are FORMAT.INV BASIC.FOR.D.T. Once you have transferred those two files to your Basic disk, boot your Basic disk and type "RUN BASIC.FOR.D.T".

If there are any further questions, please don't hesitate in writing us.

Mr. Bob Consorti

I recently received my back issue of the January Volume 1 Number 1 of **ON THREE**.

Thank you very much. It is an excellent magazine and something that Apple /// owners' have been desperately hoping to have available in previous years. Finally, some one like you, Mr. Consorti had the vision to publish the **ON THREE**.

I have been wanting to use my Apple /// to do some Fortran programming. Is there anyone that offers software that is compatible with the Apple /// computer?

If you can provide me with some names or leads I can follow up.

Thank you,

Eugene Spiewak
Michigan

Dear Mr. Spiewak:

Thank you for your enthusiastic comments about **ON THREE**. They were very much appreciated.

Appropriate people at Apple tell me that a Fortran package for the /// will be released shortly. This will be in native mode. However, if you can't wait for that, you can get the CP/M card for the /// and buy one of the CP/M Fortran packages.

I would wait, however, until the Apple version of Fortran came out.

Thank you for your continued support on **ON THREE**. If you have any further questions, please don't hesitate to write us.

Editor's Block: Continued...

products coupled with the price decrease should make everyone who has a 128K Apple /// finally upgrade their machines.

ON THREE is very proud to announce **Draw ON** /// . Described in detail on the back cover of this issue, **Draw ON** brings the power of **LisaDraw** and **MacPaint** to the Apple /// . Powerful cut and paste facilities let you create almost

anything that you can visualize. Very fast, **Draw ON** lets you "pick-up" objects on the screen and "Drag" them around in real-time. This is the most exciting package ever for the /// . With **Draw ON** you can become a one person graphic arts studio!

Since it works in any of the Apple ///'s color or B/W graphic modes and can quickly print out any picture that you create, **Draw ON** is the first program to fully take advantage of the Apple ///'s tremendous graphic capabilities. Even after using it for some time I can't believe some of the pictures that I've drawn! Complex three dimensional figures are easy to create as you can place objects in front of or behind previously drawn objects. CAD is also possible with the powerful "rubber-banding" and grids available.

Ever been bored with some of the graphs that your business charting programs create? Would you like to add different titles? Change the "Look" of the graph? Use different colors or texture black and white graphs? It's all a snap with **Draw ON** ///! Built in help screens help you master this powerful program and there is a top-quality instruction manual that rivals the beautiful books that Apple itself makes.

Over the past few months we have done demos of **Draw ON** at various trade shows and the response is almost universal - How can you do that on an Apple ///? - It wasn't easy, but **Draw ON** will truly expand your use and need of the Apple ///. Priced at only \$129, this is one of the few programs that you can't do without.

Over the past few months we have developed **ONTIME**, the **Apple /// Time-Management System**. If you have an **ON THREE O'Clock** or equivalent Clock/Calendar for your ///, you can enjoy the benefits of **ONTIME**. **ONTIME** is a software driver that permits you - the user to display at any time and from within any program, the time and date.

A simple keystroke turns on the clock display in the upper right hand corner of the screen. In the exact same format as the **SYSTEM UTILITIES** disk, **BACKUP** ///, and **Lazarus** /// **Version 2**, the time and date tick off in the "Background". Thus you can be editing a word processor file, updating some figures in a spreadsheet, or even using your Data Base Management System and the time and date will tick off second by second.

Compatible with all **Apple** /// programs running under **SOS**, **ONTIME** gives the user freedom - freedom from having to look at the clock or watch to check what time it is. With **ONTIME**, just look on your screen! Since you can be doing something else while the clock is ticking off, **ONTIME** works **with** you, not against you. After using it you'll wonder just how you got along without it.

Ever try to keep a long-distance call down to a certain number of minutes by looking at your watch and guessing when it's time to say goodbye and hang-up? With **ONTIME**, as soon as your call goes through simply turn on the **Stop-Watch** feature. The elapsed seconds will be displayed on the screen giving you an up to the second read-out of how long you've been on the phone. This timer can count up to 99 hours, 59 minutes and 59 seconds -long enough for monitoring the longest phone call imaginable.

For those of you who take your **Apple** /// everywhere, the **Stop-Watch** can even count by tenths of a second - perfect for timing your favorite horse as he races! Fantastically priced at only \$24.95, this is one of the few items that will make the **Apple** /// even easier for you to use. If you don't already have a clock, please check the special pricing on an **ON THREE O'Clock - ONTIME** combination listed elsewhere in this issue.

Editor's Block continued on page 17.

Why Is Everyone So Excited About Lazarus?

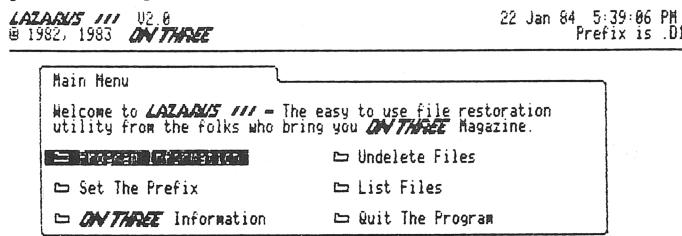
by Bob Consorti

After letter after letter saying such things as: "It's the first program that I feel like an expert at after just five minutes.", "Lazarus not only works very well, it is the best looking program I have ever seen.", and "It makes my Apple /// look and act like a Lisa!", I think it's time to explain just why everyone is so excited about Lazarus.

Lazarus /// V2.0 Unveiled

Since a picture is worth a thousand words, there are quite a few pictures here of just what Lazarus looks like. Well, they aren't actually pictures, they are screen dumps using the PKASO interface card. Since they aren't exact pictures the aspect ratio may be a bit off. In any case, when you boot the Lazarus Start-up Disk, the Lazarus Main Menu comes up in a minute or so. It is shown in Figure #1.

[FIGURE #1]

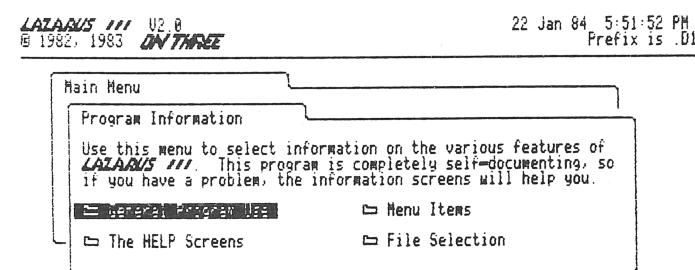


respond to the arrow keys AND by typing the letter(s) of each menu item.

In most menu driven programs, the idea of where you are in the menu is generally the same. Something called the **Highlight Bar** will overlay a menu item. In Figure #1 the highlight bar is overlaying the **Program Information** menu item.

Under most menu driven programs, if you pressed the **RETURN** key here, the **Programs Information** menu item would take over the entire screen and you could be quickly confused over just what happened to the Main Menu. With Lazarus there is no confusion! When you press the **RETURN** key, the **Program Information** menu item "Pops-Up" and "Floats" (you have to see it to believe it!) to a position overlaying part of the Main Menu. Shown in Figure #2, note that it doesn't take over the entire screen, just the part below the menu that it exists on.

[FIGURE #2]



Press: **ESCAPE** for prior menu, **RETURN** to select, ↑ ↓ ← → to move.
You may also type the name of your selection. **CTRL**-?, **d**-? or **q**-? for HELP.

Have you ever used a menu driven program, gotten lost in one of the menus, and could not find a way out? With programs as complex as the **System Utilities** and **KeyStroke**, it's not hard to get caught in one of those blind alleys and struggle for a way to break free. Actually, **System Utilities** is a complex but fairly easy program to operate. If you ever get lost in that program just press the **ESCAPE** key. This will bring you to the next higher menu.

The command to go down into a menu is usually **RETURN**. While a good approach, a novice (and even experienced users) can easily lose themselves in one of these menus and search in vain for a way out. The problem is of people either forgetting or losing track of where they are in the program. This is because the program is forcing the user to remember where he came from in the menu system.

Lazarus overcomes these difficulties by using a different type of menu. The Lazarus menu is set up as a number of electronic file cards. The TAB at the top of the first card tells you it is the Main Menu. Likewise, each other file card will identify itself in the TAB portion of the card. The contents of each file card (sometimes called - **Menu Items** -) are displayed on that file card.

On the file card, these Menu Items are normally preceded by a miniature file card. These indicate pathways into different parts of the program. Just as most menu driven programs allow you to select their menu items with the arrow keys, Lazarus will

Press: **ESCAPE** for prior menu, **RETURN** to select, ↑ ↓ ← → to move.
You may also type the name of your selection. **CTRL**-?, **d**-? or **q**-? for HELP.

The TAB portion of the Main Menu is still displayed, so that even after going down into a menu item, there is no ambiguity over where you are and how you got there. Just as in most other menu driven programs, pressing **ESCAPE** will bring you back to the next highest menu. If you press **ESCAPE** here you will get back to the Main Menu shown in Figure #1.

Now that you're at the Main Menu, press **RETURN** to "Pop-Up" the "Program Information" menu item. Since all of the Lazarus menus work in the same way, if you press **RETURN** once again the "General Program Use" menu item will "Pop-Up" as shown in Figure #3. Now press **ESCAPE** twice and you will be back at the Main Menu in Figure #1.

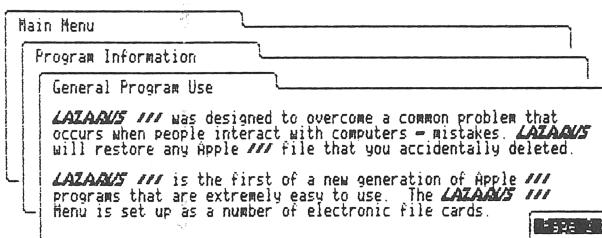
You now know how to use two of the most important keystrokes within Lazarus - **RETURN** and **ESCAPE**. Simply put, just as in other menu driven programs, **RETURN** brings you down a menu level and **ESCAPE** brings you back up.

Now that you know how to move between different menu levels, it's time to learn how to move within a menu. Starting out at the Main Menu, press **RETURN** once with the **Highlight Bar** over the "Program Information" menu item. Now press the **RIGHT-ARROW** key to move the highlight bar over the **Menu Items** menu item and press **RETURN**. The screen will now look like Figure #4.

[FIGURE #3]

LAZARUS !!! V2.0
© 1982, 1983 **ON THREE**

22 Jan 84 6:03:06 PM
Prefix is .D1

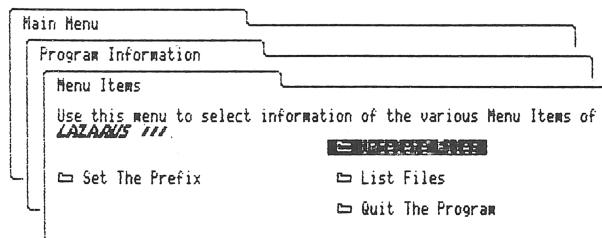


Press: **ESCAPE** for prior menu. **RETURN** to select. **↑ ↓ ← →** to move.
You may also type the name of your selection. **CTRL-?**, **d-?** or **g-?** for HELP.

[FIGURE #4]

LAZARUS !!! V2.0
© 1982, 1983 **ON THREE**

22 Jan 84 8:28:34 PM
Prefix is .D1



Press: **ESCAPE** for prior menu. **RETURN** to select. **↑ ↓ ← →** to move.
You may also type the name of your selection. **CTRL-?**, **d-?** or **g-?** for HELP.

Try the various arrow keys to see what they do. You should be able to move all around the various menus with the few keystrokes that you know. Remember, if you get "stuck" just keep pressing **ESCAPE** to get back to the Main Menu.

By now you are probably wondering just how I gave **Lazarus** the rounded edges and all the fancy looks. For those of you who think that I did everything using the graphics capabilities of the !!!, you're wrong! **Lazarus** uses the most powerful tool that the **Apple** !!! has to offer - changeable text! That's right, all of **Lazarus** is done in the standard 80 Column X 24 Line text mode. This and the fact that over 75% of **Lazarus** is written in assembly language gives the program its amazing speed and looks.

Help — ALL The Time!

The new "Pop-Up" Menus are extremely easy to use but what happens if you are stuck on something and can't remember what key to press? The answer is to simply look at the menu screen. If you will notice the bottom two lines on the screen you will see a quick description of the various keys that you can press. In addition to the keys that control movement, there are three keystrokes that will bring up **HELP** screens. The keystrokes and their effects are listed below:

CONTROL-? Lists the keys you can press and tells what each one will do.

OPEN-APPLE-? Describes the highlighted item.

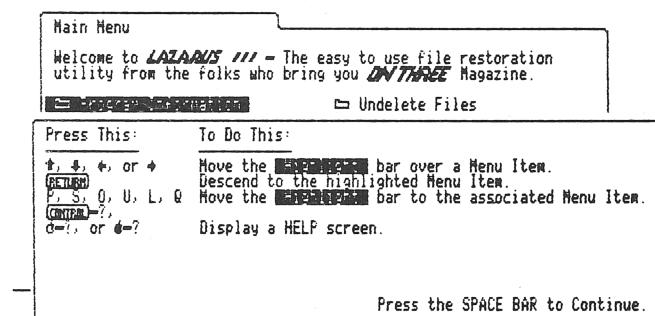
CLOSED-APPLE-? Describes the screen and its options.

At the Main Menu press **CONTROL-?** and you will be greeted with the information in **Figure #5**. Notice how it tells you not only what keys you can press, but what those keystrokes will do.

[FIGURE #5]

LAZARUS !!! V2.0
© 1982, 1983 **ON THREE**

22 Jan 84 5:40:22 PM
Prefix is .D1



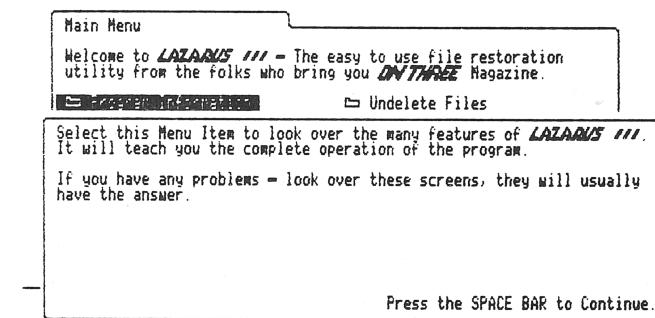
To make this help screen go away, just follow the instructions at the bottom of the screen - **Press the SPACE BAR to Continue**. This is standard through all of the **Lazarus** help screens. After playing with this for a while you may notice that instead of pressing the **SPACE BAR** you can press the **RETURN** or **ESCAPE** keys to make the help screens go away. This is a handy addition for people who don't read instructions and remember that the **ESCAPE** key is used to "Back-Out" of the current menu.

With the highlight bar over "Program Information" menu item, press **OPEN-APPLE-?** This will cause the display of **Figure #6**. This **HELP** screen tells you about the highlighted item - in this case "Program Information".

[FIGURE #6]

LAZARUS !!! V2.0
© 1982, 1983 **ON THREE**

22 Jan 84 5:41:36 PM
Prefix is .D1

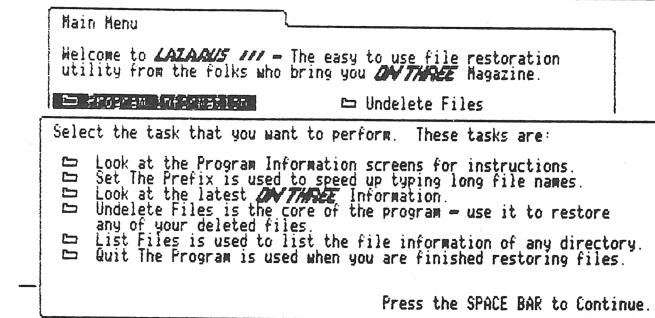


At the Main Menu press **CLOSED-APPLE-?** and **Figure #7** will be displayed. This is the help information that describes the entire screen and its options. As you can see, it gives a quick description of all the various things that **Lazarus** can do.

[FIGURE #7]

LAZARUS !!! V2.0
© 1982, 1983 **ON THREE**

22 Jan 84 5:44:03 PM
Prefix is .D1



ON THREE

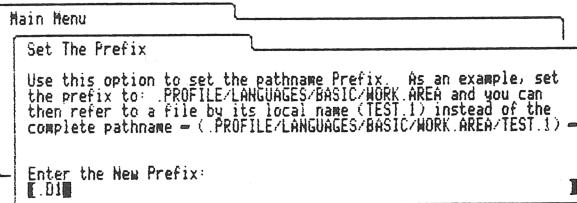
I could fill page after page with all the help screens of **Lazarus** but that wouldn't leave room for showing the other features. Just remember that wherever you are **in** the program, help is available.

Now I'd like to show you some of the things that **Lazarus** can do, and how easy it is to do them. At the Main Menu, use the arrow keys or type "S" to move the highlight bar over the **Set The Prefix** option. Now press **RETURN** and **Figure #8** will be displayed. If you are at all familiar with the **Apple //** you probably know something about the **Prefix**. This menu item allows you to set the Prefix.

[FIGURE #8]

LAZARUS !!! U2.0
© 1982, 1983 **ON THREE**

22 Jan 84 6:34:12 PM
Prefix is .D1



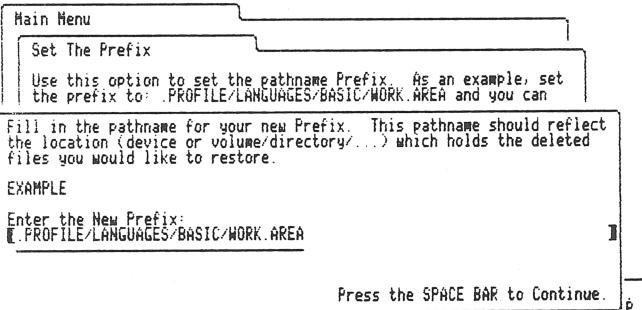
Press: **ESCAPE** for prior menu, **RETURN** to select, **↑ ↓ ← →** to move.
You may also type the name of your selection. **CONTROL-?**, **6-?** or **6-?** for HELP.

Even if you don't know what the **Prefix** is, this screen (and its associated help screens) will help you learn. Press **OPEN-APPLE-?** and **Figure #9** will be displayed. Remember that the **OPEN-APPLE-?** brings up help about the item that is highlighted. This is information that helps the user learn about the program. If this isn't enough, pressing **CLOSED-APPLE-?** will bring up **Figure #10**. This displays help about the screen in general.

[FIGURE #9]

LAZARUS !!! U2.0
© 1982, 1983 **ON THREE**

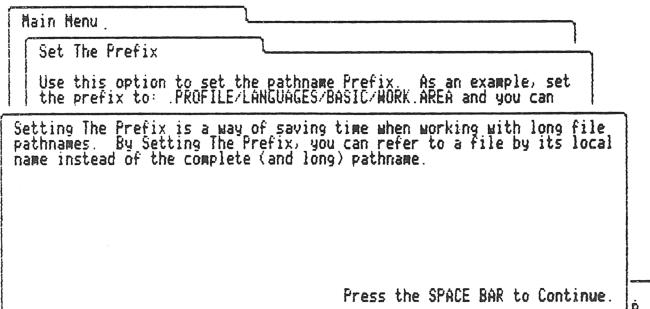
22 Jan 84 6:37:04 PM
Prefix is .D1



[FIGURE #10]

LAZARUS !!! U2.0
© 1982, 1983 **ON THREE**

22 Jan 84 6:38:24 PM
Prefix is .D1

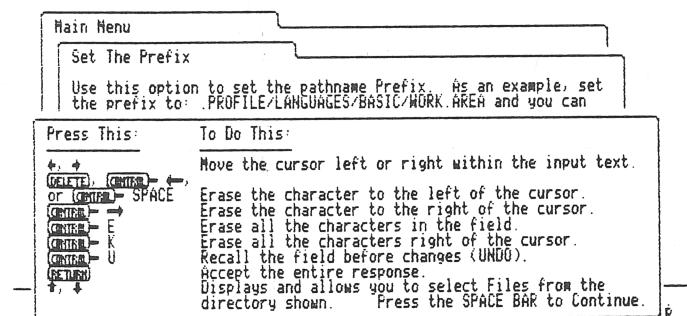


It looks easy, huh? It's even easier to work with! What happens if you want to change the prefix? Just press **CONTROL-?** to list the keys that you can press. This help screen is shown in **Figure #11**. The editing keys are from the latest Apple User-Interface Standard and the same options will be used with all **ON THREE** products in the near future.

[FIGURE #11]

LAZARUS !!! U2.0
© 1982, 1983 **ON THREE**

22 Jan 84 6:35:51 PM
Prefix is .D1

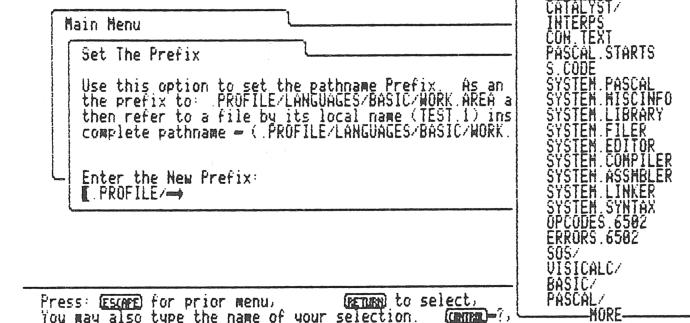


As you can see there is one very interesting option, the **File Selection** feature. Just as in the **System Utilities**, **Lazarus** allows you to press the **UP** or **DOWN** arrow keys to display a listing of the files of the directory shown on the prompt line. To illustrate, change the **Prefix** to ".PROFILE" (if you have one) and press the **UP** or **DOWN** arrow keys. A window will open up on the right side of the screen looking like **Figure #12**.

[FIGURE #12]

LAZARUS !!! U2.0
© 1982, 1983 **ON THREE**

22 Jan 84 6:41:58 PM



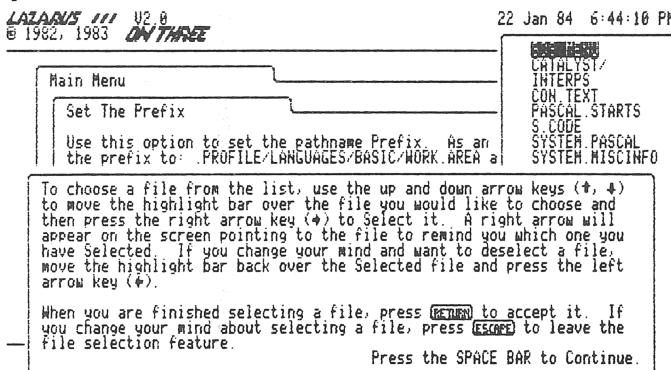
Press: **ESCAPE** for prior menu, **RETURN** to select,
You may also type the name of your selection. **CONTROL-?**,
6-? or **6-?** for HELP.

It all happens very quickly, even faster than the **System Utilities** does it! The files in the list that have a "/" after them represent sub-directories and may contain other files that you want to select. Now how do you select one of these sub-directories? You can find detailed information on doing this under the **Program Information, File Selection** screens or you can press **CONTROL-?** to give information about the keys that you can press.

This help screen is shown in **Figure #13** and tells you just how to select a file. As in most other things in **Lazarus**, this is the same throughout the program. You can select files to **Undelete** and directories to **List** in the same manner.

The "MORE" on the bottom of the window shows that there are more files in the list. If you use the **DOWN** arrow key to move the highlight bar to the bottom of the screen, the files will scroll up within the window. If you press the **RIGHT** arrow key while the highlight bar is over "PROGRAMS/" (as shown in **Figure #14**) and then press **RETURN**, you will change the prefix to ".PROFILE/PROGRAMS".

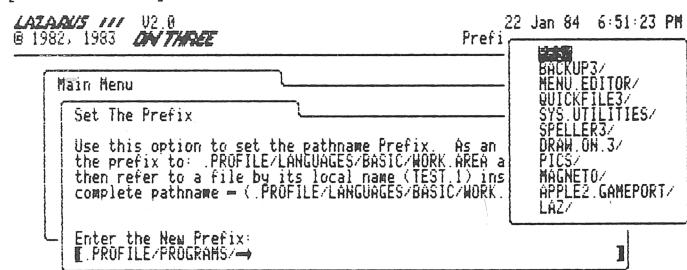
[FIGURE #13]



22 Jan 84 6:44:10 PM

MAIN MENU
CATALYST/
INTERPS
CON.TEXT
PASCAL.STARTS
S.CODE
SYSTEM.PASCAL
SYSTEM.MISINFO

[FIGURE #16]

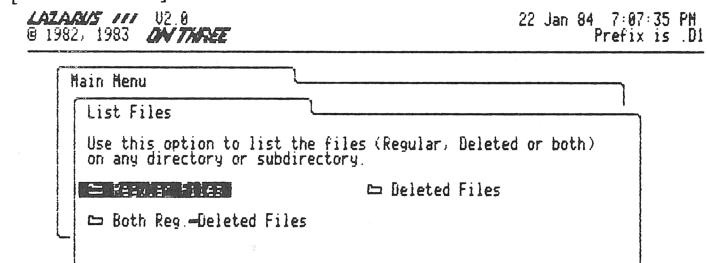


22 Jan 84 6:51:23 PM

SHCKUP3/
MENU.EDITOR/
BUCKFILE3/
SYS.UTILITIES/
SPELLERS3/
DRAW.ON.3/
PICS/
MAGNETO/
APPLE2.GAMEPORT/
LAZ/

Press: **ESCAPE** for prior menu, **RETURN** to select, \uparrow \downarrow \leftarrow to move.
You may also type the name of your selection. **CTRL**-?, **d**-? or **de**-? for HELP.

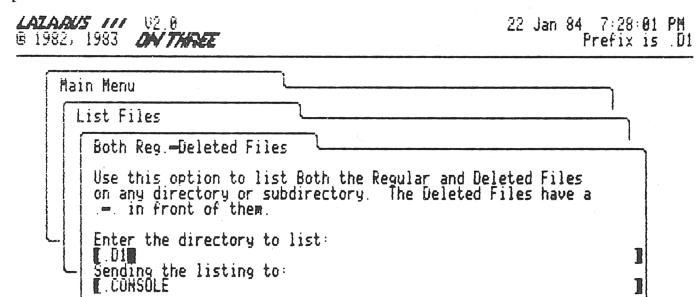
[FIGURE #17]



22 Jan 84 7:07:35 PM
Prefix is .D1

Press: **ESCAPE** for prior menu, **RETURN** to select, \uparrow \downarrow \leftarrow to move.
You may also type the name of your selection. **CTRL**-?, **d**-? or **de**-? for HELP.

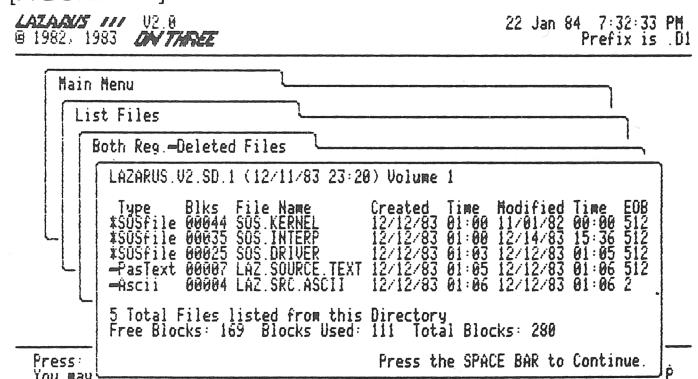
[FIGURE #18]



22 Jan 84 7:28:01 PM
Prefix is .D1

Press: **ESCAPE** for prior menu, **RETURN** to select, \uparrow \downarrow \leftarrow to move.
You may also type the name of your selection. **CTRL**-?, **d**-? or **de**-? for HELP.

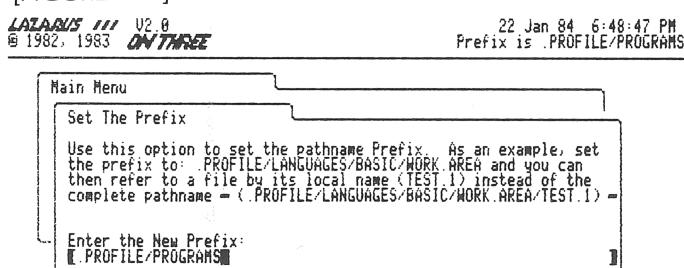
[FIGURE #19]



22 Jan 84 7:32:33 PM
Prefix is .D1

The new prefix will appear on the prompt line and the prefix displayed in the upper right of the screen will also change. This is shown in **Figure #15**. You can select files from sub-directories in the same manner as from the main directory. If you now press the **UP** or **DOWN** arrow key the files on the sub-directory **PROGRAMS** will be shown in a window. This is shown in Figure #16.

[FIGURE #15]



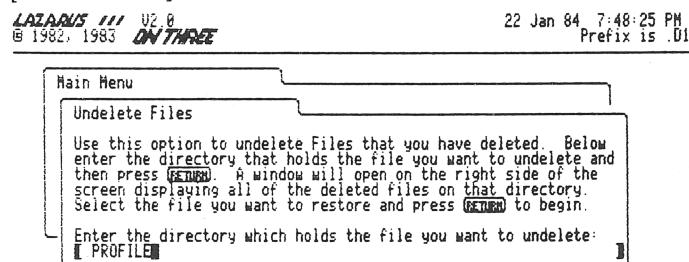
Press: **ESCAPE** for prior menu, **RETURN** to select, \uparrow \downarrow \leftarrow to move.
You may also type the name of your selection. **CTRL**-?, **d**-? or **de**-? for HELP.

Going now to some of the other options, **List Files** allows you to list any type (Regular, Deleted or both) of file on any directory or subdirectory. You can send the output to the screen, a printer or to a file on a disk. This is shown in **Figures #17-19**. The same editing and file selection features discussed before apply to this part of **Lazarus**.

The heart of the program is the **Undelete Files** option. It will restore most any file that you have accidentally deleted. Very fast, the most time it will ever take to restore a file is around 45 seconds. It can restore files from any drive connected to your ///. Regular floppy disks, hard disks such as the ProFile and even the high density Micro-Sci disk drives are all supported. If you can connect it to your /// **Lazarus** can restore files from it.

Figure #20 shows what this option looks like. As the instructions say, just press **RETURN** and a window of deleted files will appear. Use the **RIGHT** arrow key to select a file as shown in **Figure #21** and press **RETURN** to begin.

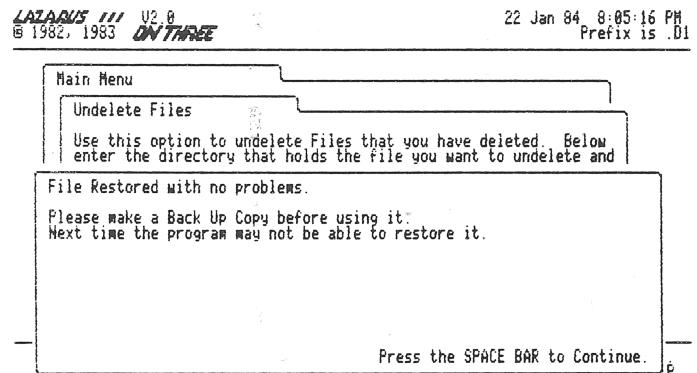
[FIGURE #20]



22 Jan 84 7:48:25 PM
Prefix is .D1

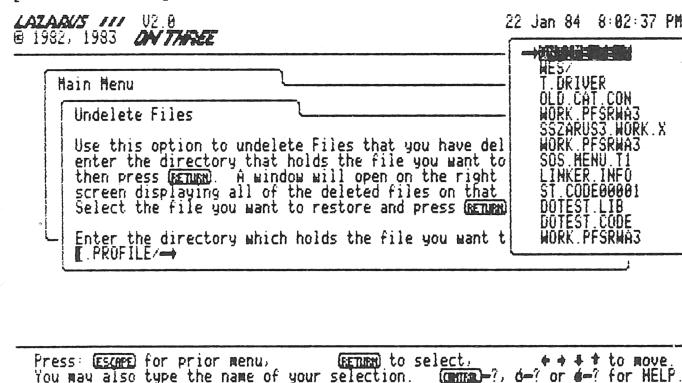
22 Jan 84 8:05:16 PM
Prefix is .D1

[FIGURE #23]



After having finished restoring your file, **Lazarus** will ask you for a name to save the file with. This is necessary due to the fact that when you delete a file, the name of the file is sometimes changed. Use the screen shown in Figure #24 to change the name back. It's that easy to restore your deleted files!

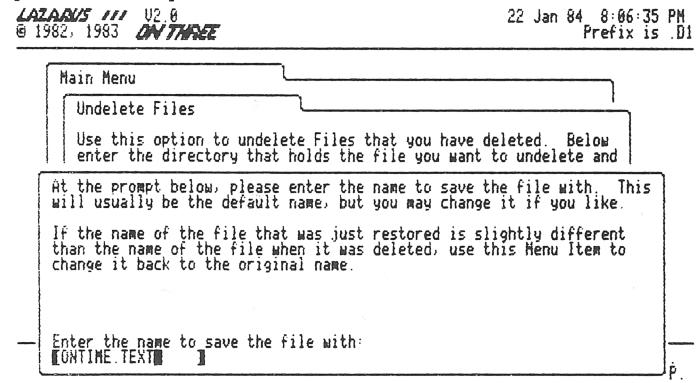
[FIGURE #21]



22 Jan 84 8:02:37 PM

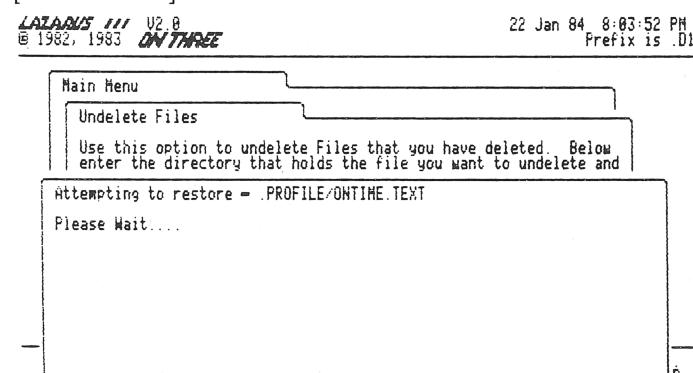
22 Jan 84 8:06:35 PM
Prefix is .D1

[FIGURE #24]



The program will then attempt to restore the file as shown in Figure #22. While the program is working to restore the file it will write a series of periods "..." across that window. This is so that the user always "Knows" that the program are working and has not "crashed" and stopped the computer.

[FIGURE #22]



I haven't mentioned it yet but one of the most important features of **Lazarus** is the **Program Information** screens. These menus will show you in detail how to use the program. We have essentially put the manual for the program **INSIDE** the program. For those of you who hate to break out the manual and page through it looking for details, with **Lazarus** you can just look through the **Program Information** screens!

Closing Notes

All of the windowing effects and very, very fast screen changes and movements were done with programming ease. The reason it was so easy was the fact that the necessary routines to do the windowing effects are built-in to the **Apple ///**.

While products such as **Visi-ON** and **Microsoft Windows** have to provide these things for the programs running under these systems, it is built-in on the ///. It's really very funny and in a way I feel sad (at least a micro-second) for the people who own PC's and XT's. You see, the programs running on those systems are not nearly as "User-Friendly" as programs like **Lazarus** (yes -

we have more coming!), but because they run in a "Window" on an imaginary "Desktop" people buy them!

These systems all consume vast quantities of memory and hard-disk space. I've heard reports that half a megabyte of main memory and up to one megabyte of disk storage space are necessary before you can put a single program onto either of these systems. That's a lot of money to lay out just so you can use a program that resides in a "Window" on your computer screen. It wouldn't be so bad if the programs worked as easy as **Lazarus** - but they don't.

Many of our future products will use the same type of displays that **Lazarus** does to make it very easy on the user. Other developers are following on the **Lazarus User-Interface** and in the future there will be many more packages available that look like **Lazarus**. In fact, the new integrated package **/// E-Z Pieces** follows the same general type of menu system of **Lazarus** though it doesn't have the pretty rounded corners and the general look of **Lazarus**.

Even after a few pages I still can't tell or show you all of the help screens and things that **Lazarus** has to offer. It's more than just a utility program, it is a new way of thinking about programs for the **Apple ///**. Many people didn't think that this type of a program could be written for the /// - that only the Lisa and the new Mac could provide that type of easy to use and spectacular looking program! Well, **Lazarus** shows that it can be done - and will be done again in the future for the ///.

Future Plans

What things are in the future for the **Apple ///** and **/// plus**? Well, **ON THREE** is working on the **Lazarus Toolkit** that will enable all future programs to use the same User-Interface of **Lazarus**. As easy to use as **Lazarus**, it will create the Menu System and Help screens for your program. With a little luck and a lot of hard work it should be available in July to developers.

While **Lazarus** and programs that use the **Lazarus Toolkit** will require a standard 256K **Apple ///** or **/// plus**, the expanded capabilities of these programs due to the Toolkit will not require massive amounts of memory and disk space. The reason: While the PC requires you to spend extra for the capabilities, the **Apple ///** has them built in! Believe it or not, we do have a leg up on the PC!

By providing many of the tools that we built for **Lazarus**, other programs will soon be able to follow this very easy and truly **User-Friendly** type of program. The net result: **Increased Productivity** for everyone using the ///. Who knows what's next - maybe someone will come out with a **Visi-ON type of system for the /// - and maybe it will be ON THREE!**

Lazarus /// is available at finer computer stores everywhere and may be ordered directly through **ON THREE** for only \$29.95. **Lazarus ///** requires an **Apple ///** with 256K or an **Apple /// plus**.

Editor's Block: Continued...

One of the other programs that you should have, **Lazarus**, is described in this issue. Read over that article and see just why people are so excited about it. Looking further into this issue, **Al Evans** is back with some more amazing things for the /// with his regular column and article **PASCALULATOR**.

Using VisiCalc for decision support is shown in **Liona Cunningham's** article while the WPL section this issue is on merging PFS and WPL files by **Jay Merritt, Kevin Fitz-Maurice** tells us why the Apple /// is the USER's Micro while **Ken Johnson** shows us how to Automate Access ///.

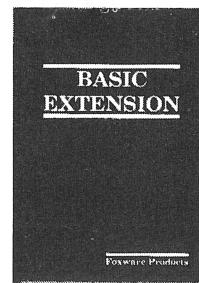
David Cortopassi tells us of the joys and tribulations of designing **CROSSWORD-SCRAMBLER**, the exciting new educational game for the Apple /// in the article So The Apple /// Is Only A Business Computer, Huh?. **Melvin Astrahan** (author of Draw ON ///) also joins our ranks this issue with the beginning of his series on Apple /// graphics.

And all this is just the tip of all the great information packed into this issue. Look over the table of contents for everything else - I can't give away everything!

For those of you who are concerned, we will do our best to publish on a more regular basis in 1984. You can help out! Please send in your article submissions today.

Next time around will be a number of reviews: Word Juggler, Apple Speller ///, BPI and more will be covered. Family Finance with VisiCalc, SOS Drivers & CP/M for those who are using that language system. We are going to attack WPL next time - From ALL angles. A Business Basic Program Lister to provide fomatted listings for those hard to read programs of yours and we will even take a deeper look at Lazarus with Pieces of Lazarus? Please stick around, there's still something for everyone! ///

BASIC EXTENSION
for the Apple ///™



If you program in Business Basic, you shouldn't be without it.

- Change file types.
- Reset protection.
- High speed disk routines, 10 times faster than Basic, up to 30% savings in disk space.
- Access to any block on a disk.
- Array manipulation, insert - delete elements, move sections of arrays, search arrays.
- Character Set Editor, create or edit your own character sets.
- Disk Block Editor, view or edit any block on a disk.
- And more, all for only \$95.

FOXWARE PRODUCTS
2506 W. Midwest Dr., Taylorsville, UT 84118
(801) 364-0394
Apple /// is a registered trademark of Apple Computer, Inc.

So The Apple /// Is Only A Business Computer, Huh?

by David Cortopassi

I'm hooked! I love my Apple ///. It's great!... It's wonderful!...

But Where's All The Software? I see no reason why I should constantly run under Apple]["Emulation". The Apple /// is capable of so much, and yet because there aren't very many ///'s out there, software houses virtually ignore it, since the market is small compared to that of the Apple][. Another blow to potential buyers of the /// was the introduction of the //e. I'm sure this was a positive move for Apple Computer (let's face it... the //e is exactly what the doctor ordered for a hungry pocketbook), but everyone is still ignoring the ///.

Just take a look at the listings of **popular** software and compare. Publications are hard pressed to limit or **select** the top applications available for the][(99% of which run on the //e). This listing usually takes up a good 10 pages, compared to 1 or 2 columns for the Apple /// (a complete list).

But, the market is growing steadily, and for the entrepreneur this could very well be the direction to go! Indeed, there are literally hundreds of computers flooding the market. But you can be sure that many (if not most) will die a quick and painful death as fast as they were designed. I'd certainly be miserable if I worked 6 months on my **dream program** in that new "FOUL" language "Fly-By-Night Computers" supplied with their B-52 compiler, only to find out they just filed for a chapter 11 (bankruptcy).

So, given this line of logic, it was only logical to commit myself to the **Apple ///**. And being a basic-plus programmer on a larger system led me to (da-daaal!) "Business Basic". Powerful isn't the word to describe "Business Basic"... I'd have to go with **Awesome. I guess that makes me a Valley Programmer.**

Where do I start? Graphics? Well, not in the true sense of the word. Although the /// has extensive graphic capabilities, I'm not obsessed by super-duper space probes emitting bursts of electro-phaser beams or what-have-you. But I **DO** get off on cursor control combined with "Normal" and "Inverse" screen display. I suppose one could think of this as a sort of "pseudo-graphics". That, combined with "WINDOW" command spurred me to undertake what turned out to be a colossal project. Which, after about 3 months of part-time dedication blossomed into **CROSSWORD-SCRAMBLER**.

Though I was not unfamiliar with data processing, I had never owned a personal computer before I purchased my Apple ///. I was disciplined in system concepts enough to formulate my ideas for crossword-Scrambler. I KNEW IT WAS PRACTICAL TO DO IN THIS LANGUAGE. This is something that I think should be decided before taking on any project. Simple experimentation showed promise and immediate reward.

I was also concerned with sound generation. With the aid of the system manuals and the Business Basic documentation, I managed to crank out the William Tell Overture in a respectable amount of time. I wasn't after laser-blast sounds, or shooting cannons; frankly, I still don't even know if it's possible to create these over-used effects with Business Basic. (Almost anything is possible - Ed.) What I wanted though, is what I got.

Between chapter 7 and appendix E of the **Standard Device**

Drivers Manual, it was a matter of editing the program example on page 131 to play a musical composition. Being a musician for the greater part of my life was an advantage here since I could relate tones to numerical values while visualizing an instrument or keyboard. I don't think it is prerequisite to have any musical knowledge though to do this, just more patience.

A good part of my interest was also in educational applications. I suppose this was partly due to the great need which exists to close the computer illiteracy gap. I also have a son who needed to learn something other than how to kill an invader from planet 6502. Thus, my vision started to take on tangible form.

I started with the initial screen display. This took a little time, as I was also after the quickest execution speed I could achieve. Section G of the Business Basic manual on 'Speeding Execution' was handy, clear cut, and to the point. I experimented by manipulating individual characters too, as opposed to printing one entire character string line. I soon found out the following:

```
10 FOR I=1 TO 40:VPOS=1:HPOS=I:PRINT
```

```
CHR$(61):NEXT
```

was slower than

```
10 A$="=====
```

```
20 PRINT AS
```

Strings and line numbers take up a lot of room. This began to matter a great deal later on when memory errors started popping up. I knew I wanted **CROSSWORD-SCRAMBLER** to fit into 128K. I also wanted the entire product to reside on one disk, and be a **TURKEY** system.

By using the System Utilities diskette I was able to open up quite a few blocks of space through reconfiguring the **SOS.DRIVER** to only recognize the built-in drive. I also deleted **.PRINTER** and **.SILENTYPE** since I had neither and **CROSSWORD-SCRAMBLER** wasn't going to need these devices. I should point out that I didn't actually delete **.D2** until I absolutely had to, as I desired to do the greater part of my programming with this capability.

Eventually, I used only the minimum system requirements for running **CROSSWORD-SCRAMBLER** as a turnkey system:

SOS.KERNEL

SOS.INTERP

SOS.DRIVER

Other Business Basic programs and data files such as RENUMBER, BGRAF, TIMESET, and extraneous fonts were also removed.

NOTE: The master disk used to make copies was created on a newly formatted disk. Each file or program was then copied individually to the new disk one by one, in a logical sequence using the System Utilities. The advantage to this method of FILE transfer rather than copying an entire disk is in making each file continuous; squeezing that teeny-tiny last bit of quality performance out of your work. It can also open up two or three more blocks if space is at a premium. This is a result of less fragmentation in the distribution of data on the disk. In the example above, **BLOCKS FREE** went from 19 to 24 after re-building the disk. All files except **score** files were then

locked as a precautionary measure. Score files are updated and thus must be unlocked.

One of many nice features about Business Basic is that it somewhat "structures" your code automatically. Unlike Pascal, where you can indent or lay out code in a definitive outline form, most basics left-justify every line. In fact, Business Basic left-justifies your code too. But, it also follows certain pre-defined rules for indentation and nesting of **FOR/NEXT** loops and such. This aids tremendously in being able to debug later on.

I finally ended up with a pretty quick screen display, and with a few minor alterations, it was able to run continually as you might see a game operating in a video parlor. After completing each section of code, I would re-program the section in order to cram as much as I could into one line number. This might seem to be bad programming structure to you, but I was more interested in saving K's than in word processing. Another reason for cramming code was that I didn't intend for the world to be able to decipher my work, and being that Business Basic runs source code means that it can easily be copied or modified to market someone else's product.

"Fantastic!", I thought. Who else was going to pat me on the back but me? I could see my vision unfolding right before my eyes. But now came the hard part, and strangely enough it had nothing to do with computers at all. I now had to make up some crossword puzzles!

I had decided on 5 different categories: Animals, Heroes & Villains, States & Countries, Anything Goes, and Music Quiz. I then bought some graph paper to figure them out. I found it much harder to make a crossword puzzle than to actually do one. This process took me almost as long as the programming. I used encyclopedias, dictionaries, comic books, a thesaurus... you name it. It was a dreadful task that literally kept me awake long into the night. I still wasn't done with words. I had to make up descriptions for each word. Spelling was another problem. "I ain't none to such good at disort a ting." You can lose a lot of credibility putting out an educational product with mistakes.

At this point, I had to write a few Q+D (quick and dirty) utility programs to create and store a data file for each subject. I kept these utilities on a separate diskette along with sound and music generator programs which I used to create songs, and various tones used to alert the player if she/he answered wrong, right, won bonus points, etc. These short subroutines could then be transferred and inserted into the desired program.

So now that I had created the puzzles, I had to get the screen to format correctly; i.e., blanking out numbers in some squares with NORMAL, while leaving those numbers that would be used to reference words either D) down or A) across. I used data statements again to accomplish this, combined with a FOR/NEXT loop this time with conditional modifier.

It wasn't easy to formulate this equation, but I found it extremely intriguing. Once I got it, it was a similar process to change the remaining puzzle formats by substituting numbers in the data statements. I determined I would break each subject down into separate programs, and run the corresponding game depending upon which subject was chosen from the main display. This is a menu-style fashion.

I found the escape key invaluable for editing and piecing programs together. In no time, I became very adept at using cursor commands as explained on page 51 of the Standard Device Drivers Manual. I'd list a sub-program, set the top of

screen to be at the bottom of the listing, load the program I wanted to append (without disturbing the screen listing display), and then reset the screen to full viewport. Next I would press escape again move the cursor to the line to insert, press escape once more, and right arrow over to the end of the line followed by a RETURN. When doing this, you must be sure that line numbers do not conflict, otherwise, you might replace a line rather than insert one.

After completing the first program (Animals), it was a matter of altering the same program to reflect each different subject. I found this easier than writing one M-O-N-S-T-E-R program. The reason being two-fold: 1) At the time, I didn't have a printer to make long listings (I was reading only from the 24X80 screen), and (2) since I wanted it to fit on a 128K system, memory size was becoming a main concern.

To eliminate the above mentioned problems, I resorted to page 246 of the Business Basic documentation on memory usage. I found this both enlightening and confusing. It took me quite a bit of time to get the proper use of the X=FRE statement. It seems Apple means this statement quite literally. I tried to assign "FRE" every variable in the book without dreaming of using "X". This eventually saved my life when repeatedly referencing array and searching data statements, especially at the second level called "SCRAMBLER".

The CLEAR statement cleared all variables including the score. The "FRE" statement reclaimed pointers without disrupting values needed for proper execution. I used the arrays to flag words that had already been put to the screen and filled into the proper place on the crossword grid. Words were listed alphabetically to speed searching with an "If greater - then return" comparison. In this way, the program could verify input and the possibility of cheating through repetition of a word more than once to rack up points.

Well, naturally there were other problems, most which appeared insurmountable. Debugging took time. I can honestly say though, that near the end I was so sick of Crossword-Scrambler I had to set it aside for a couple of weeks. Of course, my brain was still actively thinking up ways to improve upon my work. During this period I masterminded what I think to be an innovation in educational software packages: PASSWORD PROTECTION!!!

One might ask the question "why a password?" There are in fact many answers. Passwords are an everyday occurrence within the computer industry. Indeed, should a person become heavily involved with computers, they will undoubtedly use many passwords in a single day or hour. The concept of password protection and data security is one facet of which to become extremely aware. The password is therefore meant as a lesson in data security.

What a tremendous idea! Nobody ever includes a security concept in educational software, and every business uses them. Corporations spend a great deal of time, effort, and X amount of dollars on data security and encryption in an effort to prevent unauthorized access to private information. Just as you might not want your next-door neighbor to know how much money you make or owe, the idea of this can be illustrated visually. Many people aren't aware of passwords and their purpose. It seemed only natural to implement this lesson within the context of computerized educational instruction. It was an easy addition to **CROSSWORD-SCRAMBLER**.

Each game's previous score was stored on the disk at the end of program run. So, at the start of each game I gave the

user the option to see what the last score was for that particular game. Only, in order to see the previous score, you first have to give a password upon selecting that option. Of course, the password is included within the documentation, the purpose being 'example' rather than true security.

Another stumbling block worth mentioning is that the Apple /// (as does the //e) accepts BOTH UPPER AND LOWER CASE CHARACTERS! Wonderful! What this means is that when comparing character strings:

"This <> THIS"

I haD tO WRitE anotHer SuBRoutInE
to HAnDle THiS PRObleM!

Though I really felt it unnecessary due to my USER-FRIENDLY Format, the final stage was to document my work so explicitly and completely that even someone who had never touched more than a light switch could instantly feel like a computer guru. (I meant TURNKEY when I said turnkey!) I inserted on-line documentation, together with hard copy type-written information.

As I stated above, I copyrighted all of **CROSSWORD-SCRAMBLER** before making the product available. This is a simple process and the most desirable for software protection to date.

Being as I was using SOS, I needed to get Apple Computer's permission to market a product which uses their software. Apple encourages new products, and for a slight fee of only \$50 a year, they signed a contract authorizing the use of Business Basic under a company I personally own Called SOFT-PROTEX. Of course, I could have just marketed the source programs without SOS (not a bootable diskette), but that would not be a turnkey system.

To date the greatest enjoyment I have received from all my BS and Tears, is seeing my son actually select **CROSSWORD-SCRAMBLER** from an extensive software library. And, he has not only learned about each subject covered, he was right there while I was programming it! I'm amazed at his own capabilities when it comes to programming. He learned a great deal about basic by watching me. I've given him his own basic diskette, and now he asks me to run HIS programs, and get through HIS passwords! He even reads the manuals! I wish I had started that young (8 years old).

Other techniques of merit in favor of Business Basic /// are listed throughout the basic manuals. These are usually in a logical or relevant chapter. The ONLY derogatory things I can say with regard to documentation is: "Why isn't the index included in the back of both volumes 1 and 2?". This of course is extremely minor, but nonetheless was irritating at the proper moment. Better yet, why can't both volumes be combined into one?

Another thing that would be great to have would be a command that automatically puts your program into 'text' filetype format. Having to go through the trouble of opening a text file and using the LIST command with the output going to the text file, and then closing the text file, etc. is a waste of time. Why not just have another command like SAVE... only SAVETEXT.

I eventually had to make text files of all programs for submitting to the U.S. Copyright Office along with the necessary forms. As a rule I followed the sample program on page 33 of the Business Basic manual, only in immediate mode (without line numbers). This is one place where I found a small

error in Apple documentation. Line 1 in the example at the bottom of the page should read:

1 CREATE "Goodlines", TEXT

with the comma to the right and outside of the double quote rather than

1 CREATE "Goodlines,"TEXT

A "SAVETEXT" command would have been super!

I found the ON KBD statement a little mysterious, but I did manage to get it working properly. I think this and a few other commands could show more simplified examples. Still, I'm sure many would think they were straight ahead. I don't particularly like the idea of no compiler. Whether needed or not (this basic is fast), not everyone wants their source code to be public. And how do you disable the reset button or control/reset like Apple Writer ///? As far as I know, it's impossible. All I was able to do was trap Control/C and wrong type input errors. Also, I would have appreciated more concise information on SOS calls in the Standard Drivers manual. But then, I'd like to have a Quadra- Bazillion megabucks in the bank too! All and all, the Apple /// "Blows Me Away"!

As you can tell, I can't say enough about the ease of programming in Business Basic and the completeness of the documentation; not just because I'm an Apple employee, but because I'm aware of all the garbage out on the market today. The Apple /// is a superior high-technology, and judging from the looks of prototype software I see at MIS in Cupertino, Apple is committed to the same excellence for the /// as the //, //e, and LISA.

CROSSWORD-SCRAMBLER requires a 128K or 256K Apple /// with the built-in drive and monitor.

[David Cortopassi has been on the MIS Operations Staff with Apple Computer, Inc. since 1981. He is also the author of: **CROSSWORD-SCRAMBLER** - an educational software product that can be purchased exclusively through **ON THREE** Magazine.] ///

Continued from page 40.

```
READ (ACK); WRITELN;  
WRITE (CHR (1)); WRITE (CHR (28)); { Reset viewport & clear screen }  
IF (ACK = CHR(27)) THEN  
    EXIT (PROGRAM)  
END; { Of PROCEDURE Introduction }
```

```
BEGIN  
    Introduction;  
    REPEAT  
        WRITE ('Pascal UNIT NUMBER containing emulation disk: ');  
        READLN (Emul_Unit);  
        UNTIL (Emulation_Disk_Verified);  
        Patch_Character_Set;  
        WRITELN ('Your new emulation disk has been converted.')
```

Continued from page 31.

control of those pounds of notes. Or perhaps you're an attorney. Does the Apple Writer /// Database suggest a way to call up those specialized paragraphs you need for contracts? Or maybe you've been wanting to catalog all of those dozens of computer articles, and you'd like to include some notes of your own.

Or could it be you're a science fiction writer?

I Wonder what might come up if I combine all those ideas about aliens, with the ones about computers?

[L].d1/WHAT IFSI<ALIEN><COMPUTER> IA

Hmm. Just might be a story there...

#

[Sharon Webb is the author of the EARTH SONG TRIAD, published by Bantam Books, and a member of Author's Guild/League; Science Fiction Writers of America] ///

PASCALCULATOR

by Al Evans

Math on Demand in Apple // Pascal

UCSD Pascal, particularly as it is implemented on the Apple //, offers a marvelous environment for the development and use of complex software systems. However, the outer "operating-system" level cannot be characterized as user-friendly. While enormously flexible, it is rigidly structured and cannot readily be altered or expanded. Unless the system's designers included what you want to do among the command-level options, you have to write and compile a program to do it.

One difficulty I often encounter is the impossibility of typing, for example, "**PRINT 2*(289+71)**" and getting an immediate answer. This problem arises during software development or while working with my bookkeeping system or another number-manipulating program. For example, I can never remember the factors for converting degrees to radians or radians to degrees. And I may not be able, at any given moment, to figure the tax on \$73.37. But I **do** remember how to calculate 5% of 73.37 and that 2 pi radians equal 360 degrees.

Unfortunately, this knowledge is of little direct benefit in the UCSD Pascal environment. My only alternatives are to stop what I am doing and go to Basic or VisiCalc, perform the calculation, write down the answer, and reboot in Pascal, or use a calculator or pencil and paper to get the result.

I decided a routine which could be used in two ways was required: a stand-alone "pocket calculator" and a callable procedure which would, within the context of any program, evaluate a mathematical expression and return the result. For generality and ease of implementing additional operations, I chose to use real numbers for all calculations despite their limited (7 to 9-place) accuracy. Finally, since computational speed is not terribly important during keyboard input, I designed the procedure easy to modify and understand, rather than for speed.

As presented here, the unit **PASCALCULATOR** and the program **TEST_CALC** make up a stand-alone calculator which will add, subtract, multiply, divide, and exponentiate. It understands parentheses, and is not too particular about their being balanced. The unit includes a callable procedure "**Str_to_Real**" which converts real numbers expressed as strings to values of the type Real. PASCALCULATOR can be used from any Pascal program.

How To Use PASCALCULATOR

PASCALCULATOR is written as an intrinsic unit which can be placed in a library and used from any program. Enter it as shown in **Program Listing #1**, save it as "**CALC.UN**", then compile it to a codefile of the same name. The **SYSTEM LIBRARY** containing the units **TRANSCEND** and **REALMODES** must be on-line when the unit is compiled. Before an intrinsic unit can be used, it must be installed either in the system library or in a program library. Execute the utility program **LIBRARY** (originally found as **LIBRARY.CODE** on the volume /**PASCAL3**). Answer "**TESTCALC.LIB**" when the program prompts for an output file name. Answer "**CALC.UN**" when asked for an input file name. You should now see two tables, each containing 16 slots. Slot 1 on the top table contains **PASCALCU**, and the bottom table is empty. Type "**1 <SPACE>**", then "**0 <SPACE>**" (to transfer **PASCALCULATOR** to slot 0 of your new library). If you wish, you may now transfer **REALMODES** and **TRANS-**

CEND from the system library to this program library. To do this, type "N" for a new input file, enter **SYSTEM LIBRARY** for the file name, and move these units to unused slots of the output library. Now type "Q" to quit, and enter any notice you want (or just press <**RETURN**> for no notice). Your new program library is now operational.

Enter the program **TESTCALC** (**Program Listing #2**), save it as **TEST_CALC**, compile, and execute it. You should now be able to enter any numeric expression and get either a correct result or an error message pointing to the character which caused the error. If not, proofread the unit **PASCALCULATOR**. If you still can't find the problem, the procedure Calculate includes a local procedure named **Diagnose**, and a commented-out call to that procedure. Remove the brackets from around the call to Diagnose, recompile the unit, and re-install it in the library. This procedure will provide a complete dump of the operator and number stacks after each step of processing, along with the name of the next process stage and the position of the next character to be scanned. When things are working properly, comment out the call to Diagnose, recompile again, and re-install the debugged unit in the library.

The technique demonstrated in **TEST.CALC** can be used to operate **PASCALCULATOR** from any program.

How PASCALCULATOR Works

Friendly real numbers

The first problem I faced in writing this program is actually one of Pascal's best features: strong data typing. This aspect of the language is wonderful for developing well-defined programs which always behave exactly as expected. But my experience is that it has to be defeated at least once in every non-trivial program intended for use in the real world. In the present case, the problem is that UCSD Pascal is so intransigent about real numbers input from the keyboard that (1) it doesn't allow backspacing for correction and (2) any entry which is not a legitimate part of a real number crashes the system. This is totally unacceptable from the user's standpoint.

For ease of entry and manipulation, I decided that the entire expression would be entered as a string, which would then be parsed and evaluated by the supervising procedure "**Calculate**". Therefore I first wrote the procedure "**Str_to_Real**" which accepts a string and returns a real number in a designated variable. This procedure will correctly evaluate a real number in any format acceptable to UCSD Pascal. In addition, it will accept a number which begins or ends with a decimal point. It sets the boolean parameter **ERROR** to **TRUE** and the result to zero if it encounters a syntax error in the numeric string being evaluated. Like most Pascal procedures, **Str_to_Real** can be used in any program; the irritation of having to write these type-conversion routines is offset by the fact that you only have to do it once.

The main body of **Str_to_Real** extracts a mantissa sign and (if present) an exponent sign and exponent from the string passed to it. Whatever is left is considered the mantissa. **Str_to_Real** uses the local function **Pos_Real** to convert the mantissa and exponent to real values, multiplies the mantissa by the mantissa sign, and multiplies or divides the result by 10 to the (exponent) power, depending on the sign of the exponent. The procedure sets the boolean parameter **ERROR** to **TRUE** in two cases: if a portion of the input string passed to **Pos_Real** contains anything

ON THREE

```

error in expression. Values of X:
0: Successful completion
1: Invalid number
2: Number expected here
3: Operator expected here
4: No exponentiation of negative numbers
5: Cannot understand expression
6: Division by zero attempted
7: Number/result overflow
STATUS:= 100 * ERR + PRES_POS;
RESULT:= 0;
Apple /// REALMODES -- restore previous real number environment )
Rest_Env (MATH_ENV);
EXIT ("Calculate");
END; { Of PROCEDURE Error }

PROCEDURE Find_Next_CH; { Called to skip spaces }
BEGIN
  WHILE ((EXPRESSION(PRES_POS) = ' ') AND (PRES_POS < IN_LENGTH)) DO
    PRES_POS:= PRES_POS + 1;
END; { Of PROCEDURE Find_Next_CH }

PROCEDURE Perform_Operation;
VAR A, B: REAL;
  PRES_OP: CHAR;
  FUNCTION Mult_AB (A, B: REAL): REAL;
BEGIN
  Mult_AB:= A * B
END; { Of PROCEDURE Perform_Operation }

FUNCTION Divide_AB (A, B: REAL): REAL;
BEGIN
  IF (B = 0) THEN
    BEGIN
      PRES_POS:= PRES_POS - 1; { Back up one space }
      Error (6)
    END;
  Divide_AB:= A / B
END; { Of FUNCTION Divide_AB }

FUNCTION Add_AB (A, B: REAL): REAL;
BEGIN
  Add_AB:= A + B
END; { Of FUNCTION Add_AB }

FUNCTION Subtract_AB (A, B: REAL): REAL;
BEGIN
  Subtract_AB:= A - B
END; { Of FUNCTION Subtract_AB }

FUNCTION A_to_the_B (A, B: REAL): REAL;
BEGIN
  IF (A < 0) THEN BEGIN
    PRES_POS:= PRES_POS - 1; { Back up one space }
    Error (4)
  END;
  A_to_the_B:= EXP (B * LN (A))
END; { Of FUNCTION A_to_the_B }

BEGIN { Perform_Operation }
IF ((OPERATOR_STACK.POINTER < 1) OR (NUMBER_STACK.POINTER < 2)) THEN
  Error (5);
A:= NUMBER_STACK.FACTINUMBER_STACK.POINTER - 1;
B:= NUMBER_STACK.FACTINUMBER_STACK.POINTER;
NUMBER_STACK.POINTER:= NUMBER_STACK.POINTER - 1;
PRES_OP:= OPERATOR_STACK.OP(OPERATOR_STACK.POINTER);
OPERATOR_STACK.POINTER:= OPERATOR_STACK.POINTER - 1;
CASE PRES_OP OF
  '/': NUMBER_STACK.FACTINUMBER_STACK.POINTER:= Add_AB (A, B);
  '-': NUMBER_STACK.FACTINUMBER_STACK.POINTER:= Subtract_AB (A, B);
  '*': NUMBER_STACK.FACTINUMBER_STACK.POINTER:= Mult_AB (A, B);
  '/': NUMBER_STACK.FACTINUMBER_STACK.POINTER:= Divide_AB (A, B);
  '/': NUMBER_STACK.FACTINUMBER_STACK.POINTER:= A_to_the_B (A, B)
END;
END;

PROCEDURE Check_Uncary_Operator;
BEGIN
  IF (EXPRESSION(PRES_POS) IN UN_OPS) THEN
    BEGIN
      IF (EXPRESSION(PRES_POS) = '-') THEN
        UN_FACT:= -1
      ELSE
        UN_FACT:= 1;
      PRES_POS:= PRES_POS + 1
    END
  ELSE
    UN_FACT:= 1;
  PROC_STAGE:= OPEN_PAREN
END; { Of PROCEDURE Check_Uncary_Operator }

PROCEDURE Check_Open_Paren;
BEGIN
  IF (EXPRESSION(PRES_POS) = '(') THEN
    WITH OPERATOR_STACK DO
      BEGIN
        POINTER:= POINTER+1;
        IF (UN_FACT = 1) THEN
          OP(POINTER):= '('
        ELSE
          BEGIN
            { If UN_FACT is -1, combine the -1 and
              open paren into a "negative" open paren. }
            OP(POINTER):= '-'; { 'Negative' open paren }
            UN_FACT:= 1
          END;
      END;
    PRES_POS:= PRES_POS + 1;
    PROC_STAGE:= UN_OP { Set to start over }
  END;
END; { Of PROCEDURE Check_Open_Paren }

PROCEDURE Get_Number;
BEGIN
  { First character must be a digit }
  IF (NOT (EXPRESSION(PRES_POS) IN (NUMS - ['E', 'e']))) THEN
    Error (2);
  NUM_START:= PRES_POS;
  PRES_POS:= PRES_POS + 1;
  { Subsequent characters must be digits or "E" except for the character
    immediately following an "E", which can be a "+" or "-". }
  WHILE ((EXPRESSION (PRES_POS) IN NUMS) OR
    ((EXPRESSION (PRES_POS - 1) IN ['E', 'e']) AND
    (EXPRESSION (PRES_POS) IN UN_OPS))) AND
    (PRES_POS < IN_LENGTH) DO
    PRES_POS:= PRES_POS + 1;
  { Now the end of the number has been found }
  Str_to_Real (COPY (EXPRESSION, NUM_START, PRES_POS - NUM_START),
    FACTOR, REAL_ERROR);
  IF NOT REAL_ERROR THEN
    WITH NUMBER_STACK DO
      BEGIN
        POINTER:= POINTER + 1;
        FACTI(POINTER):= UN_FACT * FACTOR
      END;
  ELSE
    BEGIN
      PRES_POS:= PRES_POS - 1; { Back up one space }
      Error (1)
    END;
  PROC_STAGE:= CLOSE_PAREN
END; { Of PROCEDURE Get_Number }

PROCEDURE Check_Close_Paren;
BEGIN
  IF (EXPRESSION(PRES_POS) = ')') THEN
    WITH OPERATOR_STACK DO
      BEGIN
        WHILE (POINTER > 0) AND
          (NOT (OP(POINTER) IN ['(', ')'])) DO
          Perform_Operation;
        CASE OP(POINTER) OF
          '(': BEGIN { Negative open paren }
            NUMBER_STACK.FACTINUMBER_STACK.POINTER:=
              -NUMBER_STACK.FACTINUMBER_STACK.POINTER;
            POINTER:= POINTER - 1;
          END;
          ')': BEGIN { Cases }
            PRES_POS:= PRES_POS + 1;
            PROC_STAGE:= CLOSE_PAREN { Check for another ')' }
          END;
        ELSE
          PROC_STAGE:= GET_OP
        END;
      END;
END; { Of PROCEDURE Check_Close_Paren }

PROCEDURE Get_Operator;
BEGIN
  IF (EXPRESSION(PRES_POS) IN OPS) THEN
    WITH OPERATOR_STACK DO
      BEGIN
        WHILE (PRECEDENCE(OP(POINTER)) > PRECEDENCE(EXPRESSION(PRES_POS))) DO
          Perform_Operation;
        POINTER:= POINTER + 1;
        OP(POINTER):= EXPRESSION(PRES_POS);
        PRES_POS:= PRES_POS + 1;
        PROC_STAGE:= UN_OP
      END;
  ELSE
    Error (3)
  END; { Of PROCEDURE Get_Operator }

BEGIN
  SaveEnv (MATH_ENV); { Apple /// -- save current real number environment }
  Init;
  IN_LENGTH:= LENGTH (EXPRESSION);
  { AITow EXPRESSION [IN_LENGTH] to be addressed to simplify range checking }
  EXPRESSION:= Concat(EXPRESSION, 'X');
  PRES_POS:= 1;
  OPERATOR_STACK.POINTER:= 0;
  NUMBER_STACK.POINTER:= 0;
  STATUS:= 0;
  PROC_STAGE:= UN_OP; UN_FACT:= 1;
  WHILE (PRES_POS < IN_LENGTH) DO
    BEGIN
      CASE PROC_STAGE OF
        UN_OP: { Check_Uncary_Operator };
        OPEN_PAREN: { Check_Open_Paren };
        GET_NUM: { Get_Number };
        CLOSE_PAREN: { Check_Close_Paren };
        GET_OP: { Get_Operator };
      END; { CASE PROC_STAGE }
      Find_Next_CH;
      Diagnose; { -- Put stack dump call here for debugging }
      END; { WHILE PRES_POS < IN_LENGTH }
    { Finish up operations on stack }
    WHILE (OPERATOR_STACK.POINTER > 0) DO
      { Any open parens left on stack are meaningless; pop them off }
      IF (OPERATOR_STACK.OP(OPERATOR_STACK.POINTER) = '(') THEN
        OPERATOR_STACK.POINTER:= OPERATOR_STACK.POINTER - 1
      ELSE
        Perform_Operation;
    { If everything worked right, result is on top of number stack }
    RESULT:= NUMBER_STACK.FACT[1];
    { Apple /// REALMODES -- if overflow then return error message }
    IF (NOT FINITE (RESULT)) THEN
      Error (7);
  END;

```

conclusion no bawisq

PFS — WPL Merge

by Jay Merritt

When I first worked with **WPL** over a year ago I thought it was the greatest thing since BASIC. Mainly because of the speed of execution, searching, etc. I started using it to create utilities for many of the customers we have here at the store. I did some crazy things with it at first, just to figure it out. Then an application came up at one of the local banks which caused me to think in a different direction: **Mail List Merging**.

Now you're probably asking yourself, why didn't you use the built in merge for Apple's Mail List Manager? Well, it's a great set of routines, but it won't work on the Profile, and won't store birthdays, safety deposit boxes, dogs names or sort by zip code. I recommend **PFS:FILE/PFS:REPORT** for its simplicity, sorting, searching, and print options (like printing to disk). I also found that the typical "Girl Friday" won't scare as easily with PFS.

I realized the potential for PFS after I practiced "printing to disk" and then reading the results with Apple Writer. Good 'ol SOS allows **APPLE WRITER //** to read **DATA FILES**, as well as **ASCII FILES** (PFS creates a **DATA FILE**, **APPLE WRITER //** creates **ASCII FILES**). I studied Paul Lutus' **AUTOLETTER**, **RENUMBER**, and many of the other routines, and I figured that if I could get the hard copy to look like the file **ADDRS** on Lutus' disk then I would get the job done. What I ended up with, actually wasn't too close, but works great and can store more names than **ADDRS**, or Mail List Manager.

I thought of the layout a couple of different ways. If I used Lutus' subroutine for renumbering the address numbers, all of the addresses would have to be in memory to do so, limiting me to about 1000 names/addresses. So I decided to have the secretary enter the form numbers manually, as she enters the normal label & worry about misnumberings in the program. The first program I made was only 66 lines long, and did everything to satisfy the bank, but all information, that is the list of names, the formletter, and the program, all had to be on the same disk. Another problem was that the program didn't know when to determine the end of the file, so consequently it just ran and ran at the end - you had to hit escape when you thought all the files had been printed. If your printer ate up the paper on the 300th address, you had to either rerun the whole set, or modify the beginning program counter. So on the second version I remedied all of the above, and added default file names for the last used list file and formletter file. This is it.

Step One

Create a pfs file with at least the following:

FORM #:

END:

Somewhere in between there you might add:

FORM #:

COMPANY:

MR*MRS*MS*DR:

NAME:

ADDRESS:

CITY:

ST:

ZIP:

END:

Other fields may be included above or below this group as long as the field names aren't duplicated.

Step Two

The "**FORM #:**" field is used as a counter. When entering the labels, the FORM # will be incremented at the bottom of the screen by the PFS program. The person keying in, need only read the number & type it in. After the number, on the same line and without spacing add one character ">" for flagging and counting purposes.

After keying in several records, print them to disk with the titles. You can practice printing to paper until they look something like this:

FORM #: 33> (put a ">" next to the number)

COMPANY: Citizens National Bank

MR*MRS*MS*DR: Ms.

NAME: Marilyn Wolfe

ADDRESS: 10 W. Elm

CITY: Al

ST: II.

ZIP:

END:

FORM #: 32> (put a ">" next to the number)

COMPANY: Wayne County Bank & Trust Company

MR*MRS*MS*DR: Mr.

NAME: John Reed

ADDRESS: 215 SE Third Street

CITY: Fair

ST: II.

ZIP:

END:

FORM #: 31> put a ">" next to the number

COMPANY: Pike County Bank

MR*MRS*MS*DR: Mr.

NAME: Ike Elliott

ADDRESS:

CITY: Petburg

ST: In.

ZIP: 47455

END:

FORM #: 1> (put a ">" next to the number)

COMPANY: Crawford Co. State Bank

MR*MRS*MS*DR: Mr.

NAME: Kent Boyd

ADDRESS: 4 N. Side Square

CITY: Robins

ST: IL

ZIP: 62345

END:

FORM #: 2> (put a ">" next to the number)

COMPANY: Union Bank

MR*MRS*MS*DR: Mr.

NAME: Thomas Wuensch

ADDRESS: P. O. Box 137

ON THREE

CITY: Lions
ST: IN
ZIP: 47345
END:

The little message under the name blank is really just another field without a colon to remind the secretary to put in the flag.

Step Three

Next we need to create a formletter, like this:

April 1, 1966

(FNAME) (LNAME)
(COMPANY)
(ADDRESS)
(CITY), (ST) (ZIP)

Dear (TITLE) (LNAME),

We at Sam's lawn tractors are sure (COMPANY) can benefit from our latest in landscaping equipment. If you act quickly, (FNAME), you can get 10% off of any accessories you may need. We would love to make a delivery to (ADDRESS) some time soon.

Please call me personally (TITLE) (LNAME).

Personally yours,
Sam Paper

.ff
Notice that all of the characters inside the parenthesis are capitalized. This is to match the format used in the program below. Also the ".ff" is very important: it skips to the next page after printing.

Step Four

After that work is done the following WPL program can be DOne.

```
PND
NY
PGOBGN
FD
B
F/$C/$D/A
P
PRT
SRC
PSRCL
PSRFH
PPR ***** SEARCHING FOR FORM# (X) *****
PRT
NUM
PSRCL
PSRFH
PPR ***** WORKING ON ADDRESS (Z) OUT OF (X) SEARCHES *****
PRT
CL
PPR
PRT
HF
PPR
PRT
LN
PPR-----
PRT
ENT2
PSRLN
PPR ENTER: FILE NAME of $C & HIT RETURN
PSRLN
PRT
DF
PPR
PPRFILE NAME IS $D
PPR
PPR HIT RETURN TO ACCEPT-OTHERWISE TYPE THE CORRECT NAME
PIN ...=$C
PRT
OK
PPR
PIN IS $C OK? Y/N =$C
PRT
BN
PSX0
PSY0

PS20
PPR HAVE YOU PRINTED TO DISK? Y/N
PSRFH
PIN=$C
PCS/$C/Y/
PGOF
PCS/$C/y/
PGOF
PGOQU
F
PSRCL
PSRFH
PPR IF YOU WISH TO START WITH FORM #1 HIT RETURN
PIN OR ENTER THE FORM # TO START WITH =$C
PCS/$C//_
PGOENT
PSX$C
ENT
LSCR3
PLSSCR3&(&N=$A
PLSSCR3&(&N=$B
PFS
PSRCL
PA$mailing list (PFS print file)=$C
PSRENT2
PAS#=$D
PSRDF
PAS#=$A
PCS/$C//
PAS#=$A
PAS#=$C
PSROK
PCS/$C/N/
PGOPFS
PCS/$C/n/
PGOPFS
B
F&D&A&
Y?
AM
PSRCL
PA$form letter (APPLE WRITER file)=$C
PSRENT2
PAS#=$D
PSRDF
PAS#=$B
PCS/$C//
PAS#=$B
PAS#=$C
PSROK
PCS/$C/N/
PGOAM
PCS/$C/n/
PGOAM
B
F&D&B&
Y?
SSCR3
NY
PSRCL
L$A
B

LOOP
F&X&(&
Y?
PGOSETUP
PGOTOTAL
SETUP
PSY+1
PGO LOOP
TOTAL
NY
START
PSY+1
PSRSSC
PCS/(Z)/(Y)/
PGOOU
L$#&FORM #: (X)&END:&N
PCS/$C/$DKL/
PGOSTART
SSCR
NY
LSCRNAME: &(&N
SSCR2
NY
PS2+1
PSRNUM
L$#
PLSSCR&NAME: &(&N=$D
PAS(NAME)=$C
PSRFD
PLSSCR2# %N=$D
PAS(LNAME)=$C
PSRFD
PLSSCR&NAME: & &N=$D
PAS(FNAME)=$C
PSRFD
PLSSCR&DR: &(&N=$D
PAS(TITLE)=$C
PSRFD
PLSSCR&COMPANY: &(&N=$D
PAS(COMPANY)=$C
PSRFD
PLSSCR&ADDRESS: &(&N=$D
PAS(ADDRESS)=$C
PSRFD
PLSSCR&CITY: &(&N=$D
PAS(CITY)=$C
PSRFD
PLSSCR&ST: &(&N=$D
PAS(ST)=$C
PSRFD
PLSSCR&ZIP: &(&N=$D
PAS(ZIP)=$C
PSRFD
PNS
```

```

NY
PGO START
QU
PSR CL
PPR
PPR QUIT AFTER (Y) FORMS
PPR
PPR **** THAT'S ALL FOLKS ****
PPR COPYRIGHT 1984 - ON THREE
PPR
PPR HIT RETURN TO EXIT
PIN =$C
POT

```

Program Description

PND = NO DISPLAY

NY = NEW? YES

PGOBGN = GOTO FLAG "BGN" (BEGINNING)

BGN Routine

PS X,Y,Z = SET NUMERIC VARIABLES TO ZERO

PPR = PRINT

(I IMBEDDED A CONTROL-G JUST BEFORE THE FIRST CHARACTER OF THE QUESTION)

PSRHF = GOSUB TO FLAG "HF"

Subroutine HF

PRINT HALF A SCREEN OF BLANK LINES (VPOS 12)

PRT = RETURN FROM SUBROUTINE (GO BACK TO BGN WHERE LEFT OFF)

BGN Continued

PIN=\$C = INPUT"';CS; PAUSE AND ACCEPT RESPONSE FROM USER & STORE AS \$C

PCS = COMPARE STRINGS; IF \$C = "Y" THEN GOTO "F"

PGOQU = IF NEITHER "Y" OR "y" THEN GOTO QU (QUIT)

F Routine

.UT!

PSRCL = GOSUB TO "CL" CLEARS SCREEN WITH CONTROL-(BACK-SLASH)

.UT(BACK-SLASH)

PSRHF = VPOS 12

QUESTION = THIS IS IN CASE YOUR PAPER GOT EATEN THE FIRST TIME THRU & YOU NEED TO START WITH THE 34TH ADDRESS OR SOMETHING

PCS/\$C// = IF \$C = NOTHING ("") THEN GOTO "ENT"

PSXSC = X = VAL(\$C); STORE STRING \$C IN NUMERIC VARIABLE X; ALPHA=0

ENT Routine

LSCR3 = LOAD FILE NAME "SCR3" AT THE CURSOR POSITION IN TEXT

PLSSCR3&&(&N=\$A = LOAD STRING FROM FILE SCR3, START AT THE FIRST OCCURRENCE OF NOTHING "&&" AND STOP AT THE FIRST CARRIAGE RETURN "&(&", NOT INCLUDING THE BEGINNING AND ENDING MARKERS (NOTHING & RETURN); STORE AS \$A: SCR3 FILE HOLDS THE LAST USED FILE NAMES FOR THE PFS PRINT FILE & THE FORMLETTER FILE; \$A=PFS PRINT FILE NAME (ADDRESSES)

PLSSCR3&&(&N=\$B = SAME AS ABOVE BUT START AT THE FIRST CARRIAGE RETURN AND LOAD TO THE SECOND CARRIAGE RETURN AND STORE AS \$B: \$B = FORMLETTER FILE NAME

PFS Routine

PSRCL = GOSUB "CL"; CLEAR SCREEN

PASmailing list (PFS print file)=\$C = ASSIGN STRING; LET \$C = "XXXXXX"

PSRENT2 = GOSUB "ENT2"

Subroutine ENT2

PSRLN = GOSUB "LN"; DRAW LINE

PPR FILE NAME OF \$C & HIT RETURN = THE VARIABLE \$C WILL BE PRINTED TO THE SCREEN; CONCATENATE THE STRING MESSAGE

PSRLN = DRAW LINE ROUTINE

PRT = RETURN TO PFS ROUTINE WHERE LEFT OFF

PFS Routine Continued

PASSA=\$D = ASSIGN STRING; COPY \$A TO \$D; SET DEFAULT FILE NAME

PSRDF = GOSUB DF

DF Subroutine

PPRFILE NAME IS \$D = PRINT DEFAULT FILE NAME

PIN \$C = INPUT NEW VALUE OR NOTHING

PRT = RETURN TO PFS ROUTINE WHERE LEFT OFF

PFS Routine Continued

PASSC=\$A = WHATEVER \$C IS STORE IT IN \$A; COULD BE NULL; COULD BE FULL

PCS/\$C// = IF \$C = NOTHING THEN...

PASSD=\$A = ...PUT \$D IN THE VARIABLE \$A INSTEAD OF \$C; \$D = DEFAULT

PASSA=\$C = NOW PUT THE RIGHT FILE NAME INTO \$C

PSROK = GOSUB OK

OK Subroutine

INPUT A "Y", "y", "N", OR "n"

RETURN TO PFS WHERE LEFT OFF

PFS Routine Continued

PCS/\$C/N/= IF \$C = "N" OR "n" THEN GOTO PFS

B=GOTO THE BEGINNING OF THE FILE IN THE TEXT BUFFER (SCR3)

F&\$D&\$A& = FIND \$D (DEFAULT FILE NAME) REPLACE WITH \$A (LATEST NAME)

Y?= JUST THE FIRST OCCURENCE, THANK YOU

AW Routine

This is the same as the pfs routine except for the formletter file name.

SSCR3 = SAVE THE TEXT BUFFER AS SCR3 IN DEFAULT VOLUME; LATEST FILE NAMES

NY = CLEAR TEXT BUFFER

PSRCL = CLEAR SCREEN

This next section will count how many labels are in the print file; This requires that the file be small enough to be loaded entirely into memory, ie. less than 1100 names. This section could be replaced with a simple "PIN" requesting the number of labels manually, if there are around 1000 labels (PFS will display the final count after printing to disk).

LSA = LOAD THE LABELS FILE INTO THE TEXT BUFFER

B = GOTO BEGINNING

LOOP = MARKER; FLAG

F&>(&>(& = FIND FIRST OCCURENCE OF ">", CARRIAGE RETURN; REPLACE WITH SAME

Y?= JUST ONE

PGOSETUP = IF FOUND, GOTO SETUP

SETUP = MARKER

PSY+1 = INCREMENT Y BY 1

PGO LOOP = GOTO LOOP MARKER & DO IT AGAIN UNTIL END; GOTO TOTAL

TOTAL = MARKER; Y = NUMBER OF LABELS

NY = NEW? YES; CLEAR TEXT BUFFER

START Routine

PSY+1 = INCREMENT X BY 1

PSRSRC = GOSUB SRC

APPLE ///— THE USER'S MICRO

Some of the material in this article has already been published by "Apple Orchard" August 83', but enough is unique that I thought the readers of ON THREE might like it.

USERS VS. HACKERS

The origins of things usually tend to determine their future development as well as their audience. It is similar to the effect that a first impression often has of coloring all further experiences of the object or person in question. Microcomputers have no special protection from these human means of perception and understanding.

When microcomputers were first introduced in kit form, it was up to the hobbyist to determine a lot of the necessary activities in order to actually get the computer up and running. As time went on there appeared to be good money in providing these hobbyists with better kits, and even software, particularly the operating system software. In fact, interest seemed to be so great that it wasn't long before assembled kits were outselling do it yourself projects. With the advent of single board computers like the **Apple][**, things really began to blossom. Hobbyists, experimenters, programmers, students, and engineers were the primary consumers of microcomputers for all of these early years. We will call this original audience by the single name of hackers for convenience.

Today microcomputers come in many forms, but they are still produced with the original audience in mind. Very recently it has become popular for everyone to talk about user friendliness. Software has always had children as part of its audience, because so much of it was and still is in the form of games. So it is not surprising to find that there is indeed user friendly software available. There are even microcomputers that are as simple to operate as an eight track cassette tape, if all you want to do is to run software.

Even the newest microcomputers offer operating systems that only hackers can appreciate. An exception to this is the **LISA** from Apple. It has NO operating system (transparent) as far as the user is concerned. In fact, Apple has not even bothered to name this operating system! (How about L.O.S. 'LISA Operating System', pronounced "Loss" - Ed.) LISA is the next logical step beyond the /// in a user friendly operating system, but does not allow the access to the hacker that SOS does. Only **SOS** is both user friendly, and available to the hacker.

If you as a user want to operate the computer itself, then you must either become a hacker, or buy programs that are designed to make your operating system user oriented. DOS 3.3, PC-DOS, CP/M, TRS-DOS, Unix (Xenix), CP/M-86, MSDOS, and even UCSD Pascal are all written with the hacker and not the user in mind.

Many books have and will be published trying to make the operation of microcomputers with these operating systems possible for the general public (users). There are programs to teach CP/M, and other programs that attempt to make it user friendly. Two notable attempts to make CP/M user friendly are CP+ from Taurus Software, and Supervyz from Epic Computer Corporation. These attempts to make CP/M usable for the general public are quite valid.

by Kevin Everett FitzMaurice

What is even better than converting CP/M's cryptic commands into english like commands? NO commands at all!

THE SOS OPERATING SYSTEM

Apple computer developed a revolutionary operating system for the Apple /// and named it **Sophisticated Operating System**. The acronym winds up being **SOS**, pronounced "sauce". And for the first time we really do have HELP as this is the first operating system ever designed to be user friendly. The user need not understand SOS to use it. All the operations that a user may need, like formatting new disks or copying old ones can be found on the SOS Utilities disk. Not only are all the functions of the Utility disk menu driven, but they are cursor driven as well! Instead of commands, a user need only position the cursor with the arrow keys, and press the RETURN key. When it comes time for the user to enter information like file names, the system will prompt them for it, and even put likely names in place for them.

If you can't remember file names it is possible to look them up using the up arrow key. Help is available from the menus by pressing the Open Apple key and the question mark key. If you want to copy or destroy only certain files from a list, (subdirectory for example) you can choose which ones from the list with the up arrow. Any possible destruction of data will require confirmation, allowing you to easily change your mind or catch your error. Most operations can be aborted and menus quit by pressing the ESCAPE key. This is the easiest menu system I have ever used.

The revolution that SOS brings us continues far beyond the elimination of commands. New products for other microcomputers are just beginning to incorporate file sharing among different functions. There is talk of forming file standards to enable program file sharing. All this is good, but still way behind SOS. With SOS, ALL programs that work under SOS can share files. (I recommend that you do not buy any programs from outside suppliers that do not work under SOS.) This is a minor revolution in itself, because it allows the user to make full use of any one software package.

For example, there is no better program than a word processor for editing. Older systems have never allowed file sharing, and thus have forced people to severely limit the usefulness of their word processor. The editing functions of BASIC and UCSD Pascal are quite primitive. But with SOS and Apple Writer ///, you can edit any language including Assembler, Pascal, and BASIC. You can even use Apple Writer /// with your Visicalc ///, and Quick File /// programs. File sharing also allows the exchange of data between programs and languages. I have not yet found any limits to this truly advanced ability of SOS.

SOS also offers quite a number of other features that make it easier to use for the hacker as well. All management of memory and I/O locations and functions are handled by SOS for you. Another first for SOS is the ability to rewrite software drivers when changing peripherals, rather than having to change expensive hardware interfaces. You don't have to specify which drive a file is in, as SOS will find it for you. Also SOS, by having a built-in hierachal file structure, is the first operating system to

offer any data base capabilities. You don't have to remember which slot a card is in, because SOS will.

Files are automatically recorded with the date and time of creation, and the date and time of the last update. SOS can efficiently handle up to a 32-MEG hard disk. The first operating system from a major microcomputer manufacturer to offer user or device generated interrupts is also SOS. You can have up to eight volumes open at the same time. Pascal can communicate directly with SOS. In fact, you can use your System Utilities in place of the Pascal Filer. Storage of information is more efficient, because unlike DOS 3.3, SOS uses all available space on a disk. Programs that request printing can continue to run while the printing goes on. SOS is also unique because it is the first microcomputer operating system designed after the mainframe principles of Drivers. The Drivers (control software) for graphics and the console are a source of new power and convenience for the microcomputer.

There is much more to tell about SOS, and you can count on hearing more about it in **ON THREE**. However, for now I would like to reintroduce you to the entire Apple ///. The Apple /// was the first to offer the speed and memory ability of 16-bit chips using an 8-bit chip (6502B). When the Apple /// originally came out, it also offered the most built-in expansion ports of any microcomputer (still has more than most). Other firsts are: first major microcomputer with Winchester-type memory from its own company, and first to use 64K-byte memory chips.

FIRST OF THE RAM MACHINES

The Apple /// also offers distinct revolutionary advantages because it is the first RAM based machine. For example, the updating or changing of any language or operating system is as easy as loading a different disk. These changes then reside in RAM, as opposed to on diskette, like the slow disk based machines (some portables for example). Already this has enabled Apple to update the operating system, Business BASIC, and Pascal /// as bugs were found. This encourages Apple, or any company who copies this idea, to continually improve their products.

A RAM based machine will need a lot of RAM, and the Apple /// has a lot. An Apple /// with 256K RAM and a minimum number of drivers, can offer 205K to Business BASIC (and other!) programs. Try that on some other micros and no matter how much memory you put in their machine the operating system just can't handle it. Actually the Apple /// can handle 512K RAM, and if we ever use up all of its 256K RAM, Apple can double it (on board of course, and probably using the new 256K memory chips).

There is no more ROM than is needed for diagnostics and booting the disk (4K). This means you are not ever going to get stuck with an outdated language or system in ROM, nor will you have to work around one. Who can predict what languages we will use in the future? Who cares? Our adaptable Apple /// is ready for them now! Hopefully we will someday soon have a language for users now that we have a micro for users.

KEYBOARD

Another major concern for users is the keyboard. Typing is always a trying experience for novices. Unlike the Apple][, and even some new systems, the keyboard on the Apple /// comes in the traditional keyboard layout. What is even more important about the Apple /// keyboard is that it is completely software definable under program control at any time. This allows

multitudes of special function and help keys, and easy graphic manipulations. Two extra control keys, unique to the Apple /// (now the //e has them), allow the further redefinition of the other keys for any purpose. You get a total of 73 keys including fast numeric input ability with the numeric keypad.

What I find most revolutionary about the Apple /// keyboard is that using the SOS Utilities disk, it can be reconfigured into the **The Dvorak American Simplified Keyboard**. The Dvorak keyboard was designed for people, speed, and simplicity. The QWERTY (traditional) keyboard was designed for primitive mechanical machines above all other considerations. The keys that were pressed the most had to be kept apart on the keyboard to slow down the typing speed so the keys wouldn't jam. New users should not be forced into the insanity of the QWERTY keyboard by not being able to change it.

There are Dvorak keyboards available on typewriters from Smith-Corona and IBM, so you don't have to worry about not being able to use typewriters. Once you configure the keyboard to the superior Dvorak arrangement, it will work with all Apple /// software that runs under SOS. You can even use the Dvorak with your communications software, such as Access ///.

What makes the Dvorak so unique on the Apple /// is that it is loaded from disk every time you start the system, not installed in hardware. If you share your system with a diehard QWERTY user, it is easy to have two copies of the same program(s); one configured for the Dvorak and one configured for the QWERTY. The built-in flexibility of the Apple /// makes it truly revolutionary!

EMULATION

Apple was also the first company to offer the chance to upgrade from their old model to their new one while still retaining your old software. Granted the first Apple][Emulation disk could not run many programs, but it was a start in the right direction. The new Apple][Emulation mode is really quite good at running the /// like an Apple][or][+. When the next and perhaps final Apple][Emulation (//e Emulation?) program comes out it should be able to emulate a souped up Apple][.

If you are thinking about upgrading, then you should know what the /// will run. Let it suffice you to know for now, that if a non-game program will run on an Apple][+ with 48K and DOS 3.3, it will run on your Apple /// whether it is in Integer BASIC, Applesoft, or 6502 Assembly. This covers thousands of the programs available to the serious user.

You game aficionados however are in trouble. It is quite possible to play games on an Apple /// of course, but problems arise with those that use or can use joysticks. The keyboard is presently the only way to use Apple][games, and games that have a joystick option often won't even load. Cursor /// joysticks on the Apple ///, so far, only work in Apple /// mode. Our basic rule is, if it is meant for keyboard use it will run on the ///. But as you all know from reading **ON THREE** you can add a card from **Micro-Sci** that will completely correct this and allow all Apple][game programs to work in Emulation mode.

BUILT-IN HARDWARE

And now a list of the built-in features of the Apple ///.

INTEGRATED FEATURES.

- 1) An Internal 143K Disk drive.
- 2) Interface for up to FOUR disk drives.
(The THREE external drives can hold up to 572K)
- 3) B/W Text Mode: Full 80 columns by 24 lines.

- /// /// /// /// /// /// /// /// /// /// /// /// /// ///
- 4) Color Text Mode: 16 colors with 40 columns by 24 lines.
- 5) RGB or any video device, NTSC, and RCA phono monitor ports.
- 6) RS-232-C port standard.
- 7) Color Graphics. Full 16 colors with 192 lines of 280 dots per line.
- 8) B/W Hi-resolution Graphics. 192 lines of 560 dots per line.
- 9) Audio output. 64 volume settings and over seven octaves.
- 10) Real time clock.
- 11) Two Joystick ports, A & B. Silentype printer can share port A.
- 12) Diagnostics in ROM. Automatic booting of disk at power on.
- 13) Built-in security mount to prevent theft.
- 14) 512K RAM ability.
- 15) Four expansion slots.

KEYBOARD FEATURES.

- 1) Dvorak Keyboard layout available on file.
- 2) STRICTLY traditional typewriter layout standard.
- 3) 73 keys total.
- 4) Sculptured, tiered, and angled keys.
- 5) Two system keys. Closed and Open Apple keys redefine others.
- 6) Dedicated cursor keys.
- 7) 13 KEY Numeric key pad.
- 8) Numeric keypad can be redefined into 13 special function keys.
- 9) Alpha lock key. Shift locks alphabetical keys only.
- 10) Two keys to cold boot a disk without powering on and off.
- 11) Up to 13 control keys.
- 12) Completely software definable character set.
- 13) All symbols used by programming languages are available.
- 14) Two speed automatic repeat for all keys.
- 15) Total of 224 printable characters.
- 16) Little bumps on D, K, and 5 keys for home positioning.
- 17) 128 byte type-ahead buffer.
- 18) Many different fonts available.
- 19) Character set can be changed under program control at any time.

SUPPORTED EXPANSION PRODUCTS.

- 1) Universal parallel interface card
- 2) Monitor ///, 16 shades of green, full graphics (RCA phono port).
- 3) Silentype inexpensive printer plugs in port A.
- 4) CP/M adapter made by Microsoft.
- 5) Profile 5-MEG super quiet hard disk drive. Fits under monitor.
- 6) Qume Sprint 5 letter quality printer (plugs into RS-232-C).
- 7) Cursor /// joysticks plug into ports A and B.
- 8) Modem eliminator comes with /// for use of RS-232-C with printers.
- 9) Prototyping board for development of new boards.
- 10) Most Apple][expansion boards with an Apple /// driver.

FINAL NOTE

Apple seems to be alone in the realization that with every passing day there are more users of microcomputers than hackers. (Others are aware - Ed.) It should be obvious that the percentage of owners who are hackers will continue to drop dramatically until hackers are all but forgotten. This being the case, when it is determined what is revolutionary, it can no longer be based on what the hacker appreciates as revolutionary, but what is revolutionary for the user.

You may have read in magazines that IBM has introduced a

revolutionary microcomputer. Unfortunately, because of microcomputer's origins, most editors of computer magazines are entrenched hackers surrounded by hackers, and conditioned to evaluate products from the hackers viewpoint. To the user it makes no difference if the microprocessor has a superior instruction set. What matters to the user is how much memory can actually be used. Even speed is not a factor the user is ever likely to need worry about. I have yet to see a slow system from the user's viewpoint (Diskette based systems are slow!). Professionals who need a lot of scientific number crunching or time sharing need to worry about speed. And if they do, they will find that there are faster 8-bit systems out there than some new 16-bit systems.

This so-called revolutionary system offers the user an unfriendly primitive hacker's operating system, no more than 64K in BASIC regardless of how much memory you put in, NO built-in ports, even color is not built-in, nor are graphics built-in, it has less expansion room than much older systems, continues with all the old ROM problems, offers little speed improvement if any, has a noisy fan, has a noisy keyboard, takes up a lot of desk space, has a base of only four colors in high resolution, and comes locked into a non standard keyboard (both shift keys and the return key are out of place). And if that wasn't enough, I also hear complaints it has at least one bug in its BASIC, and needs two separate monitors (one for graphics and one for text), but can't run two. If the Apple /// doesn't win over this name brand throwback, then we might as well hand the Japanese the market.

A real challenger to the Apple /// is the Grid Compass computer. This system has some true technological advances, like 256K bubble memory, a real 16-bit chip (8086), an 8087 arithmetic processor, an 8089 I/O processor, built-in 300/1200 baud modem, and a built-in flat screen display all in a neat little package. However it also has such flaws as a small 6" screen, a normal display of only 52 characters, no batteries, no color, no sound, only 57 keys, and an outrageous price (\$9,000). But far more serious, and even a little terrifying, is that if Big Brother were to design a computer this would be it. Why? Because this is a completely closed computer system.

The company who makes it is the only source of programs, and there is no access to the insides of the computer except through a network run by the same company. Using this system, large companies now have the ability to completely control what is allowed to be done with a personal computer. If you use the Compass with the network as it is designed to be used, then these companies can monitor everything you do. They and only they can decide what programs you can get, and what information you are allowed to receive. If the Apple /// has a real enemy, a real opposite, than this is it. Someone better get TRON on this right away! After all the hero of the movie TRON used an Apple /// at home. Did you notice?

Why has the truly revolutionary Apple /// not been recognized as revolutionary? Because it has a super refined 8-bit chip instead of a "revolutionary" 16-bit chip. If the first microcomputer with a 16-bit chip is the hacker's only criterion for what is revolutionary, then hackers should be praising the TI-99/4A.

Continued from page 41.

an important plus for some kinds of work. And as a "report writer," it can be extremely versatile.

But chances are you're not in real estate. What about other uses? Well, maybe you're writing a doctoral thesis. A little imagination in creating specialized fields is all you need to get

Continued on page 20.

Automating Access //

by Ken Johnson

Access //, Apple's terminal emulation program for the Apple //, can provide you with an excellent means to communicate by phone with electronic bulletin boards and other computer systems. If you are considering Access // for your **electronic communication** needs, you can explore the merits of the program yourself with your Apple dealer. If you are a relatively new user of Access // or even if you have been using it for some time, this article will show you ways you can put it to work more effectively.

Before we start, let's look at just what form Access // is in when you get it and what you have to do, or have someone do for you, to make it work. Version 1.1 is the latest revision, incidentally, and you should obtain this update from your dealer if you have the earlier version. I have heard that a version 2.0 is in the works, but as of this writing it has not yet been released. Access // leads a double life, so to speak. It lives one life as a **Business BASIC invokable module** and the other as a **Pascal executable code file**. Both versions are assembly language code files and work exactly the same once you get them "configured" and up and running, but the method of doing this is different for BASIC than for Pascal.

Apple gives you instructions and provides a rudimentary Business BASIC **HELLO** program that does little more than INVOKE and PERFORM the Access // invokable module. It gets you going, but little else. They also tell you how to set up the Pascal version as **SYSTEM.STARTUP** and get you going from Pascal. With a little imagination, and given an autodial modem such as the Hayes Smartmodem, you can automate such things as dialing up the system you are "accessing" and perform a few other little useful chores along the way. Programmed automatic dialup of a distant computer by simply selecting it from a menu can avoid the embarrassment and expense of dialing wrong numbers and perhaps having a human rather than a computer answer. If you are calling long distance and want to take advantage of one of the services such as MCI or SPRINT to save money, automatic dialup can be even more of a convenience. The listings presented here are intended to provide a means of doing this and may spur your imagination to bigger and better things as well. We'll proceed on the assumption that Access // is to be configured to a single-stage boot disk with the volume name /**APPLCOM** in line with the instructions provided in the manual.

Let's start with the BASIC version and try a little more exotic HELLO program than Apple's version. **Program Listing #1** is our modified HELLO program. Before we jump into the listing, let's prepare our disk. Boot the System Utilities disk and format a blank disk. Don't name the disk /APPLCOM just yet. The Access // master disk has that volume name already, and the file transfer option of System Utilities will not transfer files between disk volumes if both disk volumes have the same name. Let the Utilities program assign its own default volume name for now. Transfer **SOS.KERNEL**, **SOS.INTERP**, **TIMESET.INV**, and **TIMESET** onto this disk from your Business BASIC master disk using the file transfer option. Now transfer **SOS.DRIVER** and **ACCESS3.INV** from your Access // master disk, after which you can rename your disk volume /APPLCOM. This should give you a driver file consisting of the **SILENTYPE**, **CONSOLE**, and **RS232** drivers. You can use the System Configuration Program option to replace **SILENTYPE** with your particular

printer driver at this point, if you wish. Reboot with this disk and you should see the familiar Business BASIC) prompt. Now enter **Program Listing #1**, save it as HELLO, and we are almost done.

Now, let's look at the program. Line 20 introduces something which may be new to you, the unpublished reserved variables **DATES** and **TIMES**. You might be thinking that since no values have been assigned to these variables yet, they should be blank strings, right? Wrong! **DATES** and **TIMES** are reserved variables which return the date and time values current in the elusive Apple // system clock. Even if you do not have the clock chip installed, you can set this clock, which is what we are about to do, and it will automatically date and time stamp any files you may be recording. Try entering ?**DATES** and ?**TIMES** in immediate mode from BASIC sometime, after you have run the **TIMESET** program and set the date and time. Then, save a file to the disk and you will see how the file is date and time stamped to the date and time you set. A peculiar feature about **DATES** is that it displays the year first. I suppose there is a reason for this, but I'm not sure just what it is.

On with our listing. If the current date is okay, execution proceeds to statement 50. A "N" or "n" entry will run the **TIMESET** program and let you set the system clock. We transferred **TIMESET** and **TIMESET.INV** from the Business BASIC master disk, remember? Some modifications to this copy of **TIMESET** are needed. To do this, load **TIMESET** and change line 9999 of **TIMESET** to read:

9999 TEXT:HOME:INVOKE:RUN"HELLO"

Then delete lines 60 and 179, unlock **TIMESET**, since it was transferred as a locked file, and save the modified version in its place. This modification runs your **HELLO** program again after you have set the date and time, clearing **TIMESET.INV** from memory in the process. The other modifications delete line 60, which opens the audio driver, and line 179, which produces a cuckoo clock sound effect using the audio driver. The audio driver is not used by Access // and is not really necessary to set the system clock either. Reducing the size of your driver file may be critical, particularly if you are using a 128K Apple //, in order to avoid '**OUT OF MEMORY**' errors when you go to **INVOKE ACCESS3.INV**. The .CONSOLE and RS232 drivers are the only essential ones to Access //. A printer driver is a good one to have too, but not essential unless you want to use the printer as a recording file.

The selection menu is easily expanded with a few simple rules. The "Enter Terminal Mode Without Dialing" and "Exit to BASIC" options should probably be kept as the last two on the list as others are added. Extend the range of numbers in the IF statement in line 120 and the statement list in line 130 accordingly as more options are added to the menu. PRINT CHR\$(30) in line 120, if you do not recognize it, is the console driver screen control code for **CLEAR LINE**. More on the use of these later.

The string format shown for dialing the different systems is for the Hayes Smartmodem. The phone numbers just as you would dial them are placed in the string right after ATD. You could add automatic logon sequences, access codes for MCI, SPRINT, or other reduced rate long distance calling services, and timed automatic redials as well, if desired, with a little ingenuity. The Smartmodem will accept up to 40 characters in a single

command string. The GOTO 160 branch instruction is needed after each dial instruction on the list except the last.

Line 160 PERFORMs ACCESS3 and branches execution back to the menu upon exiting from ACCESS3. The menu selection process will repeat itself as many times as you want until the "Exit to BASIC" option is selected from the menu. This option branches to line 170 where various housecleaning chores are performed. You are then put back into BASIC with a "clean slate".

At this point, you should have a single-stage boot disk which will have about 50 blocks, depending on the size of your SOS.DRIVER file, available for recording files. Applewriter ///, or a similar word processing program which generates ASCII text files, can be used to create text files for later transmission by Access ///. These files can be placed on the /APPLCOM volume as space permits or kept on another disk in drive 2.

Now that we have seen how to automate the BASIC version of Access ///, let's look at how to do much the same thing with the Pascal version. ACCESS3.CODE is the same assembly language module in function as ACCESS3.INV, but is pre-linked instead to a small Pascal host program. It can be Xecuted from Pascal Command Level or renamed SYSTEM.STARTUP to run automatically when booted. Setting up a bootable disk for Pascal is similar to BASIC, but the source of the files is different. SOS.KERNEL, SOS.INTERP, SYSTEM.PASCAL, and SYSTEM.MISCINFO are transferred from PASCAL1. SOS.DRIVER and ACCESS3.CODE are transferred from the Access /// master disk. This can be done using either the System Utilities program or Pascal Filer.

Pascal's normal method of setting the system clock uses the D)ate option of the Pascal Filer. We'll do it a different way so that we can automate it from within our program, and in the process look at such things as **SEGMENT PROCEDURES** and the use of library **UNITS**. We'll also throw in an example of how to use **UNITWRITE** to pass screen control commands to the Apple ///'s console driver. **Program Listing #2** is a procedure we'll call **Set_Clock** for checking and setting the system clock. It is a general purpose procedure that can be used with any Pascal program. **Program Listing #3** is our STARTUP program, Access_Startup. The compiled code version of this listing should be saved to your Pascal configured /APPLCOM disk as SYSTEM.STARTUP.

Set_Clock is written as a SEGMENT PROCEDURE. There is no need to keep it in memory after you have used it. It uses two built-in procedures which are part of the **APPLESTUFF** library unit, one called CLOCKINFO to read the clock and the other called SETTIME which sets the system clock. These procedures are covered in the Pascal manual. The APPLESTUFF unit must therefore be declared at the beginning of any host program using this procedure as we have done in Access_Startup. PROCEDURE Set_Clock contains a **BOOLEAN FUNCTION**, **Clock_Set** which returns the value TRUE once we are satisfied that the clock is set properly. The body of the PROCEDURE consists entirely of a **WHILE** loop which calls this FUNCTION and resets the clock as long as a value of FALSE is returned by the FUNCTION.

Let's look now at our Access_Startup program. The **USES** declaration tells the compiler which library units to link into the program. As we said, the APPLESTUFF unit is needed for the intrinsic PROCEDURES CLOCKINFO and SETTIME. The **CHAIN-STUFF** unit is also needed since we use its SETCHAIN PROCEDURE to execute ACCESS3.CODE within our program. Both units must be present in a file named **SYSTEM LIBRARY**

on our program disk. Use the Pascal LIBRARY.CODE program found on PASCAL3 to create this SYSTEM.LIBRARY file on the /APPLCOM volume. SEGMENT PROCEDURE Set_Clock is inserted where the comment indicates. By definition, such procedures must precede any other executable code in the program.

Next is a short PROCEDURE we'll call **Home**, since it does the same as HOME does in BASIC. It illustrates the use of UNITWRITE to pass control commands directly to the console driver. We've used UNITWRITE slightly differently further on, just to show that there is more than one way it can be done. These codes are found, incidentally in the Standard Device Driver's Manual under the section on the CONSOLE driver.

Access_Startup is relatively simple and straightforward. The listing is documented to show each step. It uses a **REPEAT** loop to read in the menu selection, repeating the loop **UNTIL** the selection falls within the integer set 1..4 of the menu. The **SOCHECK-** option is used here to disable Pascal's automatic I/O checking and to prevent the program from terminating with an I/O Error upon entry of the wrong input type, a letter for example. To add to the options, simply add the additional selections to the menu and **CASE** statements and expand the range of the 'selection' set accordingly. Once a valid selection is entered, the program proceeds to the **CASE** statement that selects the proper string value for the string variable, dial. Dialing the number is simply a matter of writing the value of the variable, dial, out to the Smartmodem. The SETCHAIN procedure then actually exits this program, chaining execution of ACCESS3.CODE.

Unlike the BASIC version, we do not return directly to Access_Startup after we exit from Access /// with the option of making another menu selection. We return instead to the Pascal command level. If the code file is named SYSTEM.STARTUP, you need enter only I)initialize to re-run Access_Startup. This may seem somewhat more awkward than the way it is done from BASIC, and it is. To make it as convenient would require that ACCESS3.CODE be furnished as a library unit, linkable into a host program. Perhaps in future versions of Access /// it will be.

Since both versions of Access /// work the same, why have two? The BASIC version seems the more versatile of the two and leaves more room on the /APPLCOM disk for other files. It is also much easier to modify the BASIC HELLO program than the Pascal SYSTEM.STARTUP program, since the Pascal program must be recompiled after each change. Does the Pascal version have any advantages over the BASIC version? I have found that the Pascal version actually offers a great deal more flexibility and convenience. The Pascal Editor and Pascal Filer are directly accessible from Pascal command level. The System Utilities program, itself a Pascal code file, can also be made easily accessible directly from Pascal command level without having to boot it separately. Here's one way that can be done.

Take a second formatted disk and name it /APPLCOM.D2. Transfer SYSTEM.FILER and SYSTEM.EDITOR to this disk from PASCAL1. Then transfer the SYSTEM.STARTUP file from your System Utilities disk to /APPLCOM.D2, renaming it UTIL.CODE. (This procedure is explained in Appendix I of the Standard Device Drivers Manual.) Keep this disk in drive 2 when running the Pascal version of Access ///. Now you can enter the Editor or Filer just by selecting the E)ditor or F)iler option from Pascal command level. SOS will seek out and load the appropriate files from drive 2 automatically. You can also run the System Utilities program by Xecuting .D2/UTIL from the Pascal command level. You can use all of its features, including formatting blank disks.

Just make sure that the FMTD driver is part of the SOS.DRIVER file on your boot disk, if you want to use the disk formatting option. This second disk has the additional benefit of providing more disk space for recording files, about 41 blocks in fact. Your single-stage Pascal boot disk for Access /// will only have about 14 blocks left at this point.

Having the Pascal Editor conveniently available makes it very easy to create text files for transmission by Access ///. Version 1.1 of Access /// will transmit either Pascal or ASCII text files. The Apple /// Pascal Editor can generate either and has nearly all of the features of a dedicated word processing program. I have found it even better suited to generating text files for transmission to bulletin board systems than Applewriter ///. The Pascal Editor will automatically insert the carriage return at the end of each line. Most bulletin board systems seem to require this.

As we said, Applewriter /// can be used to create ASCII text files for transmission by the BASIC version of Access ///. The System Utilities program can also be used in conjunction with the BASIC version. These require separate boots, however, a somewhat more awkward process. The convenience of having the Pascal Editor and Filer and System Utilities program so easily accessible from Pascal gives this version the edge in many applications. Both versions have their strengths and weaknesses, but furnished in the form that they are, they comprise a powerful and remarkably versatile data communications system.

Access ///:

Program Listing #2

```
SEGMENT PROCEDURE Set_Clock; { Only in memory when called }

VAR setting : STRING [14]; { Input string for new clock setting }
newssetting : STRING [18]; { Reconstructed string for SETTIME procedure }
day : STRING [1]; { Not CHAR since CONCAT requires STRING type }

FUNCTION Clock_Set : BOOLEAN;
  VAR year, mon, daycount, dayofwk, hr, min, sec, thou : INTEGER;
    { Values returned by APPLESTUFF CLOCKINFO procedure }
  okay : CHAR;
BEGIN { FUNCTION Clock_Set }
  CLOCKINFO (year, mon, daycount, dayofwk, hr, min, sec, thou);
  { APPLESTUFF procedure call reads the current clock setting }
  WRITELN; WRITELN ('System date and time are as follows:');
  WRITE ('Day:'); { Break it down into readable form }
CASE dayofwk OF
  1 : WRITELN ('Sunday');
  2 : WRITELN ('Monday');
  3 : WRITELN ('Tuesday');
  4 : WRITELN ('Wednesday');
  5 : WRITELN ('Thursday');
  6 : WRITELN ('Friday');
  7 : WRITELN ('Saturday');
END;
WRITELN ('Date: ', mon, '/', daycount, '/', year);
WRITE ('Time: ', hr, ':');
IF (min < 10) THEN { Make it easier to read }
  BEGIN
    WRITE ('0');
    WRITELN (min);
  END
ELSE
  WRITELN (min); { Write it as is }
WRITELN;
WRITE ('Press (RETURN) if okay, N to change: ');
READLN (okay);
IF ((okay = 'N') OR (okay = 'n')) THEN { Check both upper and lower case }
  Clock_Set := FALSE { We want to reset it }
ELSE
  Clock_Set := TRUE; { Current setting is okay }
END; { FUNCTION Clock_Set }

BEGIN { SEGMENT PROCEDURE Set_Clock }
  WHILE (Clock_Set = FALSE) DO { Call FUNCTION Clock_Set and test it }
    BEGIN { Reset the clock }
      day := '0';
      WRITE ('Enter the day as: Sunday = 1... Saturday = 7: ');
      WHILE ((day < '1') OR (day > '7')) DO READLN (day);
      WRITE ('Enter date and time as MM/DD/YYYY HH:MM: ');
      READLN (setting);
      newssetting := CONCAT (COPY (setting, 7, 4), COPY (setting, 1, 2),
        COPY (setting, 4, 2), day, COPY (setting, 12, 2),
        COPY (setting, 15, 2), '00000');
      SETTIME (newssetting); { APPLESTUFF procedure call }
    END; { WHILE }
END; { SEGMENT PROCEDURE Set_Clock }
```

Program Listing #3

```
PROGRAM Access_Startup;
{ **** AUTOMATING ACCESS ///: PASCAL VERSION ****
{ * Copyright 1984 by | |
{ * ON THREE | |
{ * by Ken Johnson | |
{ * Volume 2, #1 | |
{ * |
{ * This program is a modified SYSTEM.STARTUP for the Pascal version
{ * of Access ///. It will autodial the numbers that you supply.
{ * |
{ * Please read the accompanying article for complete information.
{ * |
{ * **** AUTOMATING ACCESS ///: PASCAL VERSION **** }

USES CHAINSTUFF, APPLESTUFF; { Must be in SYSTEM LIBRARY (or Program Lib.) }

VAR selection, { Menu selection }
  code : INTEGER; { Screen control code variable }
  dial : STRING; { For telephone numbers }
  smartmodem : INTERACTIVE;

{ INSERT SEGMENT PROCEDURE Set_Clock HERE }

PROCEDURE Home; { Works same as HOME in BASIC }
VAR code : INTEGER;
BEGIN { Home }
  code := 28; { Clear Viewport }
  UNITWRITE (1, code, 1);
END; { Home }

BEGIN { Access_Startup }
  Home; SetClock; { Set the system clock }
  RESET (smartmodem, '.RS232'); { Open the modem as an INTERACTIVE file }
  Home;
  GOTOKY (10, 5); { And print the menu to the screen }
  WRITELN ('Welcome to Access ///');
  WRITELN;
  WRITELN ('Which system do you wish to contact?');
  WRITELN;
  WRITELN ('1 - First Computer System');
  WRITELN ('2 - Second Computer System');
  WRITELN ('3 - Enter Terminal Mode Without Dialing');
  WRITELN; WRITELN;
  selection := 0;
  REPEAT { Until valid selection made }
    code := 31;
    GOTOKY (5, 20); UNITWRITE (1, code, 1); { CLEAR TO END OF LINE }
    WRITE ('Enter your selection: ');
    {$IOCHECK=0} { Disable automatic checking }
    READLN (selection);
    {$IOCHECK=1} { Re-enable automatic checking }
    UNTIL (selection IN [1..4]); { Expand set range when adding to menu }
CASE selection OF
  1 : dial := ATD(First Computer System phone number);
  2 : dial := ATD(Second Computer System phone number);
  3 : dial := '';
  4 : BEGIN
    Home;
    EXIT (PROGRAM);
  END; { Back to command level }
END; { CASE }
WRITELN (smartmodem, dial); { Dial the number }
SETCHAIN ('/APPLCOM/ACCESS3'); { Execute Access /// }
CLOSE (smartmodem); { Never really executes, but good programming practice }
END. { Access_Startup }
```

Program Listing #1

```
REM **** AUTOMATING ACCESS ///: BASIC VERSION ****
REM * Copyright 1984 by | |
REM * ON THREE | |
REM * by Ken Johnson | |
REM * Volume 2, #1 | |
REM *
REM * This program is modified HELLO program for the Basic version
REM * of Access ///. It will autodial the numbers that you supply.
REM * Please read the accompanying article for complete instructions.
REM **** AUTOMATING ACCESS ///: BASIC VERSION ****
TEXT:HOME
20 PRINT "DATE:PRINT" Time is: "; TIME$;
30 PRINT "Press (RETURN) if okay, N to change: "; GET ok$: IF INSTR("Nn", ok$)
  THEN RUN#TIMESET$;
40 OPEN#1, ".RS232";
50 a$= PREFIX$+PREFIX$=":INVOKE"/APPLCOM/ACCESS3.INV"
60 HOME:HP0$=30:INVERSE:PRINT" WELCOME TO ACCESS /// :NORMAL
70 VP0$=3:HP0$=22:PRINT" Which system do you wish to contact? "
80 VP0$=6:HP0$=26:PRINT"1 - First System"
90 VP0$=8:HP0$=26:PRINT"2 - Second System"
100 VP0$=12:HP0$=26:PRINT"3 - Enter Terminal Mod Without Dialing"
110 VP0$=15:HP0$=26:PRINT"4 - Exit to BASIC"
120 (selection() OR (selection()4)) THEN PRINT selection:PRINT selection:IF
  (selection() OR (selection()4)) THEN PRINT CHR$(30):GOTO 120
130 ON selection GOTO 140,150,160,170
140 PRINT#1,ATD(First System phone number)":GOTO 160
150 PRINT#1,ATD(Second System phone number)"
160 PERFORM ACCESS3:GOTO 60
170 PREFIX$=a$:INVOKE:HOME:NEW:END
```

Slots of Fun

by Ben McCaw

Dear Sir:

My name is Ben McCaw. I am 10 years old. I drew the pictures for the enclosed slot machine program, that uses the **Apple** // window command to make the screen look like a real slot machine. My father helped me with the programming.

Lines 10 to 64 are the heart of the program. They use two loops to print my faces at random in the five different windows. Lines 100 to 670 are my faces for the slot machine. You could have any faces that you want. Lines 900-1301 keep track of your money and winnings.

I would like to see the program printed in **ON THREE** and I think your readers will enjoy seeing how easy it is for a kid to make neat graphics.

Sincerely,

Ben McCaw

Dear Ben,

I think you're right and I'm sure our readers will like to see this program.

Bob Consorti

```

0 REM ****
1 REM *
2 REM * Slots Of Fun:           Copyright 1984 BY *
3 REM * (Ben's SUPER Slot Machine)   ON THREE *
4 REM * ----- Volume 2, #1 *
5 REM * by Ben McCaw *
6 REM ****
7 M=10.00:N=M-.25
8 TEXT:HOME:WINDOW 30,2 TO 75,3:PRINT"Ben's SUPER Slot Machine"
9 GOSUB 900
10 FOR T=1 TO 7
39   P=1
40   FOR I=1 TO 60 STEP 12
41     WINDOW 1,9 TO 10+I,20
42     R=INT(RND(2)*35)
43     R(P)=INT(R/6)+1
44     ON INT(R/6)+1 GOSUB 100,200,300,400,500,600
45     P=P+1
46   NEXT I
47   NEXT T
48 GOSUB 1000
49 WINDOW 10,22 TO 30,24
50 INPUT"PRESS RETURN TO PLAY";A$
51 GOTO 4
52 PRINT"2222222222"
53 PRINT"? ?"
54 PRINT"S 0 0 S"
55 PRINT"S S"
56 PRINT"S 00 S"
57 PRINT"S S"
58 PRINT"S S"
59 PRINT"S W W S"
60 PRINT"S WWWW S"
61 PRINT"S S"
62 PRINT"S SSSSSSSSS"
63 RETURN
64
100 PRINT"2222222222"
101 PRINT"? ?"
102 PRINT"S 0 0 S"
103 PRINT"S S"
104 PRINT"S 00 S"
105 PRINT"S S"
106 PRINT"S S"
107 PRINT"S W W S"
108 PRINT"S WWWW S"
109 PRINT"S S"
110 PRINT"S SSSSSSSSS"
111 RETURN
200 PRINT"AAAAAAA"
201 PRINT"A A"
202 PRINT"A II A"
203 PRINT"A AAAA A"
204 PRINT"A AAAAA A"
205 PRINT"A AAAAA A"

```

```

206 PRINT"A AAAAAA A"
207 PRINT"A AAAA A"
208 PRINT"A AA A"
209 PRINT"A A A"
210 PRINT"AAAAAAA"
211 RETURN
300 PRINT"
301 PRINT"=====
302 PRINT"=====
303 PRINT"
304 PRINT"
305 PRINT"=====
306 PRINT"
307 PRINT"
308 PRINT"=====
309 PRINT"=====
310 PRINT"
311 RETURN
400 PRINT"BBBBBBBBBB"
401 PRINT"B II B"
402 PRINT"B BB B"
403 PRINT"B BB B"
404 PRINT"B BB B"
410 PRINT"B BB B"
420 PRINT"B BB B"
430 PRINT"B BB B"
440 PRINT"B BB B"
450 PRINT"B BB B"
460 PRINT"BBBBBBBBBB"
470 RETURN
500 PRINT"CCCCCCCCC"
501 PRINT"C SS C"
502 PRINT"C SS C"
504 PRINT"C S C"
505 PRINT"C S C"
510 PRINT"C S S C"
520 PRINT"C OO S C"
560 PRINT"C OO OO C"
561 PRINT"C OO C"
570 PRINT"C C"
580 PRINT"CCCCCCCCC"
590 RETURN
600 PRINT"GGGGGG-G-G"
601 PRINT"GGG GGG"
602 PRINT"GG GG"
603 PRINT"G O O G"
604 PRINT"S S"
605 PRINT"S S"
606 PRINT"S S S S"
607 PRINT"S SSSS S"
660 PRINT"S S"
665 PRINT"S S"
670 PRINT"SSSSSSSS"
900 WINDOW 62,24 TO 79,24:PRINT"YOU HAVE $":PRINT USING"###.##";M;
901 RETURN
1000 W=0
1010 IF R(1)=3 AND R(2)=3 AND R(3)=3 THEN W=40:GOTO 1200
1020 IF R(1)=3 AND R(2)=3 THEN W=5:GOTO 1200
1100 IF R(1)=R(2) AND R(2)=R(3) THEN W=16:GOTO 1200
1110 IF R(1)=R(2) THEN W=2:GOTO 1200
1120 IF R(2)=3 AND R(3)=3 THEN W=3:GOTO 1200
1121 IF R(3)=3 AND R(4)=3 THEN W=3:GOTO 1200
1122 IF R(4)=3 AND R(5)=3 THEN W=3:GOTO 1200
1130 IF R(2)=R(3) OR R(3)=R(4) OR R(4)=R(5) THEN W=1.50:GOTO 1200
1200 M=MW
1205 IF W>0 THEN GOSUB 1300
1206 IF W>0 THEN PRINT CHR$(7)
1207 GOSUB 900
1210 RETURN
1300 WINDOW 63,22 TO 79,23:INVERSE:PRINT"YOU WON $";W:NORMAL
1301 RETURN

```

/// to the Max #3 (Would You Believe][to the Max?)

by Al Evans

OK, OK, I know I said I would talk about making sounds on the Apple /// in this column. But that will have to wait until next month, because this month I found something you'll like better.

One thing I've always found irritating about the Apple /// is the "**Apple][Emulation Mode**". Not that it doesn't emulate the Apple][— it does that very well in most respects — but after all, it IS an Apple ///. Surely there must be some way to take advantage of some of the more powerful features of the REAL machine without losing compatibility.

In particular, it seems downright unfair that the emulation mode offers only upper-case letters. I haven't had an Apple][in years without a lower-case adapter. But there's no place to put a lower-case adapter on the ///, or a shift-wire modification, or any of the other bits and pieces with which we old-timers "customized" our Apple]['s. And besides, the Apple /// is already supposed to HAVE upper and lower case letters. Isn't there some way we can get them into the emulation mode?

As it turns out, there is. An intrepid explorer named George Oetzel has boldly ventured into the unknown depths of the Apple][Emulation disk. He has discovered emulation modes far beyond anything an Apple][ever dreamed of, and his series of articles in **Softalk** (June, July, and August, 1983) is well worth reading. However, we will content ourselves for now with a program which will convert an emulation disk to allow any Apple /// system character set to be used in the emulation mode in both upper and lower case. As an added attraction, you can choose either a flashing or a non-flashing cursor.

There are several steps which must be taken. First, of course, a new character set must be moved onto the emulation disk. The accompanying program begins by reading a standard Apple /// character set. Then it reads the appropriate track from the emulation disk which is to receive the new Apple][character set, moves the new character set into place, and writes the track back to the emulation disk. In itself, this would allow any Apple][program to display lower-case letters, but would not permit them to be typed in from the keyboard.

The next step involves making a few short patches to the Apple][monitor program — to allow it to accept lower-case input and keep it from producing strange effects when the cursor is backed over a letter. Since the entire emulation mode, including the system monitor, loads into RAM on the Apple ///, this is much easier than altering the ROM monitor on a real Apple][.

The third step is to provide a new KEYIN routine for the monitor, one which takes the status of the SHIFT and ALPHA LOCK keys into account before deciding which character has been entered. George Oetzel has written such a routine, and all Apple /// owners should thank him for it. It is patched into a portion of the monitor originally used to read cassettes. Since the /// has no cassette port anyway, this will cause no problems.

The program makes these changes in the same way as before, with one additional complication — two copies of the Apple][monitor exist on the emulation disk, at different offsets from the beginnings of their disk tracks. For this reason, the procedure which actually performs the patching (Make_

Patches) requires an OFFSET into the track buffer as a parameter.

Finally, we have to change things so that Applesoft never tries to go into FLASH mode — since we now have lower-case letters, this will produce garbage on the screen. Again, the Applesoft language actually loads into RAM on the ///, so we can make these modifications directly on the disk image.

How to use the program

To use the accompanying **A2PATCH** program, you need 2 pieces of information. First, you have to know the **Pascal unit number** of the disk drive in which you will place the emulation disk to be modified. This is necessary because the emulation disk has no directory, and can only be read and written by absolute block addresses. If you have a 2-drive system, the built-in drive is unit #4 and the second drive is unit #5. If you have a ProFile, things are somewhat more complex. The easiest way to be certain is to press "F" at the Pascal command line to get to the filer, then press "V" for "volumes on line". This command shows the Pascal unit numbers, device names, and, where applicable, the volume names for all devices configured into the system.

The second piece of information you need is the **pathname** of the file containing the character set you wish to use. Any standard Apple /// character set is acceptable. Several are available on the Apple Business Basic disk and the System Utilities DATA disk. More can be found on the Apple Writer /// Master disk. Or, you can design your own font using any of a number of available character set editors (including my /CHARSET/EDITOR).

Of course, you also need a copy of your Apple][Emulation disk to modify. Place this copy in the drive you will use and execute the program. Enter the Pascal unit number for this drive when prompted. The program checks to verify that an emulation disk is in the specified drive. Now enter the pathname for your character set file. The program loads the character set, checks again to be sure the emulation disk is still present (and demands that you replace it if it's not), then installs the character set and makes the required patches. The whole process takes about 5 seconds.

Boot the Apple /// on your new emulation disk, then place any bootable DOS 3.3 disk in the built-in drive and press <RETURN>. It should be possible to enter and display caps and lower-case letters with no further ado. As mentioned above, the FLASH command in Applesoft no longer works. The reason is rather complicated, but basically it had to be removed to allow inverse lower-case.

If the effect you get is not what you expected, proofread the program text VERY carefully, particularly the procedures Set_Keyin_Patch, Make_Patches, and Patch_Applesoft. The numbers in these procedures all represent absolute locations and machine-language instructions. In that realm, there is no such thing as a "slight" mistake.

Incidentally, this process has one side-effect I don't understand. If, and only if, you issue a DOS "FP" or "INT" command, the cursor temporarily changes to a non-flashing inverse **accent grave** (ASCII character 96). It changes back to the

normal inverse or flashing space character when the next key is pressed.

And onward . . .

I can think of a couple of improvements to this program, and I'm sure you can think of more. For example, it's irritating to have to refer to a disk drive by its Pascal unit number. Unfortunately, a SOS device name can be translated to a unit number only by means of a two-stage process involving a SOS call, and thus a linked-in assembly-language routine. For the sake of time and space, I left it out. Of course, a procedure to provide a catalog of any disk suspected of containing character-set files would also be useful.

It would be nice to be able to access more of the Apple ///'s features from emulation mode. As George Oetzel explains in the articles cited above, this is possible — to some extent. Changes can be made in the program which sets up the emulation mode, for example to make it possible to both read from and write to the RAM from \$D000 to \$FFFF, to switch the \$C000-\$CFFF address space between RAM and I/O, to introduce a color text mode, and to allow the clock to be read. But these changes become increasingly complicated and make the emulation mode increasingly unlike a real Apple][— and, hey, if you just want to PLAY with a computer, well the Apple /// is already the greatest toy there is!

Back on a more reasonable level, you could change the NMI and RESET vectors in the monitor (at locations OFFSET + 762 through OFFSET + 765) to point to location \$FF59 (fill those locations with 89 255 89 255), relieving the application program of its control over what happens when you hit RESET.

Or, for example, you could patch Applesoft so that it actually reads the Apple /// joystick when PDL(0) and PDL(1) are referenced. Since my joystick is in port B, Applesoft normally reads it as paddles 1 and 3, which means I have to change any program I want to use with it. You can patch Applesoft itself to jump from the routine that normally handles the PDL keyword to some part of the code which isn't used (for example, the part of the language which processes STORE, an ancient keyword which had something to do with storing arrays on cassettes). Then insert a short routine at that location to change 0 to 1 and 1 to 3 before JSR'ing to PREAD. Unfortunately, there's nothing you can do about the fact that the two buttons on that joystick are still read as Apple][buttons 1 and 3 — and the Apple][doesn't even HAVE a button 3!

And then some things are just impossible. For example, there's apparently no way to fake the Apple /// into believing it has a language card. And my personal dream of an Apple][running INSIDE an Apple /// will apparently go unrealized. Ah, but what a computer THAT would make!

Anyway, that's it for this column. I'd like to call your attention to the fact that the patcher program has only one main procedure, Patch_Character_Set. Lots of logical space left for other interesting and/or useful alterations. Send them to me, and I'll include them in a future column. As I may have mentioned before, we've got to SHARE OUR KNOWLEDGE!

```
PROGRAM A2PATCH;
{ **** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * }
{ * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * }
{ * * Apple ][ Patch * Copyright 1984 by I * }
{ * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * }
{ * * by Al Evans (with help from George Oetzel) | Volume 2, #1 | * * * * * * * * * * * * }
{ * * This program will alter an Apple ][ Emulation diskette so that * * }
```

```
{ * you can install any Apple /// character set. You can now also * }
{ * have upper AND lower case letters in Emulation mode! * }
{ * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * }

TYPE Byte = 0..255;
Block = PACKED ARRAY [0..511] OF Byte;
Track = PACKED ARRAY [0..4095] OF Byte;
SYS_C_Set = PACKED ARRAY [0..1023] OF Byte;

VAR T_BUF: Track;
ERUL_UNIT: INTEGER;

PROCEDURE Read Track (TRACK_NO: INTEGER);
VAR START_BLOCK: INTEGER;
BEGIN
  START_BLOCK:= TRACK_NO # 8;
  UNITREAD (EMUL_UNIT!, T_BUF, sizeof (T_BUF), START_BLOCK, 12)
END; { Of PROCEDURE Read_Track }

PROCEDURE Write Track (TRACK_NO: INTEGER);
VAR START_BLOCK: INTEGER;
BEGIN
  START_BLOCK:= TRACK_NO # 8;
  UNITWRITE (EMUL_UNIT!, T_BUF, sizeof (T_BUF), START_BLOCK, 12)
END; { Of PROCEDURE Write_Track }

FUNCTION Emulation_Disk_Verified: BOOLEAN;
VAR BLK: Block;
INDEX: INTEGER;
CHK_BUF: PACKED ARRAY [0..7] OF Byte;
BEGIN
{ Check against the commands which actually set up Emulation Mode }
CHK_BUF[0]:= 93; CHK_BUF[1]:= 227; CHK_BUF[2]:= 255; { LDA $0FEF3 }
CHK_BUF[3]:= 9; CHK_BUF[4]:= 64; { STA $A400 }
CHK_BUF[5]:= 141; CHK_BUF[6]:= 227; CHK_BUF[7]:= 255; { STA $0FFE3 }
UNITREAD (EMUL_UNIT!, BLK, sizeof (BLK), 2, 12); { Read in block 2 }
FOR INDEX:= 0 TO 7 DO
  IF (CHK_BUF[INDEX] < BLK[383 + INDEX]) THEN
    BEGIN
      Emulation_Disk_Verified:= FALSE;
      EXIT (Emulation_Disk_Verified)
    END;
{ If we get here, the bytes checked out }
Emulation_Disk_Verified:= TRUE
END; { Of FUNCTION Emulation_Disk_Verified }

PROCEDURE Patch Character_Set;
VAR C_Set: SYS_C_Set;

PROCEDURE Get_CSet;
VAR C_FILE: FT1e OF SYS_C_Set;
PATHNAME: STRING;
IO_RES, INDEX: INTEGER;
CURSOR_TYPE: CHAR;
BEGIN
  WRITE ('Pathname of character set file: ');
  READLN (PATHNAME);
  REPEAT
    ($10CHECK-)
    RESET (C_FILE, PATHNAME);
    ($10CHECK+)
    IO_RES:= IORESULT;
    IF (IO_RES = 0) THEN
      C_SET:= C_FILE
    ELSE
      BEGIN
        WRITELN ('Error in reading ', PATHNAME, ', (IORESULT = ', IO_RES, ')');
        WRITE ('Pathname (RETURN) to exit program: ');
        READLN (PATHNAME);
        IF (Length (PATHNAME) = 0) THEN
          EXIT (PROGRAM)
      END
  UNTIL (IO_RES = 0);
  WRITE ('Flashing or Solid cursor? (F or S)');
  READ (CURSOR_TYPE);
  WRITELN;
  FOR INDEX:= 256 to 263 DO { Locations of bytes defining SPACE character }
    CASE CURSOR_TYPE OF
      'F', 'f': IF (C_SET[INDEX] < 128) THEN
        C_SET[INDEX]:= C_SET[INDEX] + 128;
      'S', 's': IF (C_SET[INDEX] > 127) THEN
        C_SET[INDEX]:= C_SET[INDEX] - 128;
    END;
END; { Of PROCEDURE Get_CSet }

PROCEDURE Patch Monitor;
VAR KIP: PACKED ARRAY [0..48] OF Byte; { KEYIN PATCH }
{ These patches for the Apple ][ emulation monitor
to allow lower-case letters are by George Oetzel. }

PROCEDURE Set_Keyin_Patch;
BEGIN
  KIP[0]:= 132; KIP[1]:= 47; { STY 43 }
  KIP[2]:= 134; KIP[3]:= 66; { STX 42 }
  KIP[4]:= 173; KIP[5]:= 0; KIP[6]:= 192; { LDA $0C000 }
  KIP[7]:= 168; { TAY }
  KIP[8]:= 173; KIP[9]:= 0; KIP[10]:= 192; { LDA $0C008 }
  KIP[11]:= 170; { TAX }
  KIP[12]:= 44; KIP[13]:= 16; KIP[14]:= 192; { BIT $0C10 }
  KIP[15]:= 41; KIP[16]:= 8; { AND #08 }
  KIP[17]:= 208; KIP[18]:= 0; { BNE +6 }
  KIP[19]:= 152; { TYA }
  KIP[20]:= 166; KIP[21]:= 66; { LDX 42 }

```

Program listing continued on page 40

Decision Support With VisiCalc

by Llona Cunningham

Learning VisiCalc commands is the first step in making use of the program. Transferring to the screen what we used to do by calculator is the next step. But to put this powerful program's potential to work, you need to get involved with template design as an end product.

There are more factors in template design than could be covered in a single column; certainly more than most of us have time to digest in a single list of instructions. The best way to learn is still, **by doing**.

While in VisiCalc, whether studying an example template, or creating one of your own, begin with understanding what is needed from the information available. In this example, a weekend pilot needs the bottom line cost of an airplane with expected revenue and all the variable use-costs.

When Is An Airplane A Good Investment?

Our weekend pilot has been approached by the local rental/charter service to buy an airplane for his own use and rent it back to the service. The premise, of course, is that if he owns the plane he can fly for considerably less than for his current rental fee (on a per/hour cost basis). Renting the plane back to the service would make the capital outlay a tax shelter investment, while letting the rent collected make the loan payments. And the ownership worries could be kept to a minimum by the charter service handling all the little necessary details (for a small fee of course). The costs of meeting regulation requirements could be scheduled by the number of hours used. And on, and on.

Coming from the Sales rep it sounds too good to pass up. In fact, it sounds too good to be true!

As the weekend pilot peaks with interest, so does the list of unknowns and their effect on the bottom line. Like, exactly what are the costs associated with "x" number of hour's use? What kind of savings could be had over the simple rent-by-the-hour fees? And what would the bottom line look like over a period of five years? (Five years is the maximum loan period for a craft of that age.)

The list of variables associated with the deal include:

1. Capital Investment

The combined total of down payment, loan amount and initial out-of-pocket costs. Down payment and out-of-pocket costs are included with the loan amount on the assumption that they would otherwise remain in savings. The forfeited interest which would have been earned on savings is then "included" as part of the interest "paid" on the capital investment.

2. Years

The period of the loan, which is also a good measure of probable "investment life" for depreciation purposes, helps establish a time factor for our weekend pilot to balance against the other variables.

3. Interest and % Annual Percentage Rate

Most loans of this type are not simple interest ($x\%$ times full amount times full number of years). Declining interest ($x\%$ times a declining balance) comes to about half the simple interest fee. Amortized loans and "Rule of 78's" financing, are an altogether different matter. For this template, the Declining interest method is assumed to accommodate a rough average of the interest paid on a loan and the interest forfeited by withdrawing from savings.

4. Owner Tax Bracket

This can be either the actual tax percentage (the percent of actual taxes paid from actual income at year-end) or the new dollars tax bracket (the percent of tax owed on new dollars).

5. Owner Hours / Week

Estimation of the average hours per week the owner would expect to fly whether in his own plane or through rentals. Since even the owner of the plane must pay his share of incidental use-costs, it was established that he would pay a discounted rental fee at time of use (the rent fee minus the commission normally retained by the rental service).

6. Hourly rental fee

The hourly fee charged to renting customers (assumed to be the same as that paid by the weekend pilot if he doesn't buy the plane).

7. Owner's Percentage

A percentage of the collected fees goes to the service as commission fees, the remaining percent to the owner to meet expenses (and/or net a profit).

8. Average fee hrs/mo

Average number of hours per month in which paying customers will be renting the plane. (Owners use is accounted for in #4)

9. Fuel cost per gallon

The price of fuel per gallon changes from month to month as well as from airport to airport. Since all expenses including fuel are paid for from the owner's portion of rental income, the cost of fuel can have a deciding effect on the fairness of the rental fee as well as on the bottom line investment value for the owner.

10. Avg. Fuel per hr

In most cases, this is figured as cruising speed consumption. Add 10-20% if much flying is done over mountainous terrain, or if student flying and ground maneuvers make up a large part of the use.

11. 50 hr. check

The fee charged by the flight service station to make a customary inspection (and perform maintenance as needed) at mid-point between 100 hour intervals.

12. 100 hr. check

The fee charged for the 100 hour inspection and estimated normal maintenance.

13. 1800 hr. check

As above, but this one usually entails major engine overhaul and is best kept to minimum by making sure the 50 and 100 hour maintenance procedures are followed religiously.

14. Insurance / mo

Owner's insurance for commercial-use craft, divided into monthly installments.

15. Misc. / mo

The template is designed for single screen width (standard 8.5" wide paper). The top half is for variable input. The bottom half for calculation. Entries displaying "—0—" are user-input cells. The actual algorithms used in computation are listed and explained below.

Note that the interest figure in cell E4 displays "0.00" — this is a computed cell based on the interest figure entered in cell F4. Do not enter the interest dollar figure in E4 unless the financing is based on a finite dollar amount rather than interest percentage.

Computed cells in the bottom half will display "error" until numbers are entered above. In the blank template, those cells are trying to divide by 0, which always produces an error message.

Airplane Investment Template Construction

Decision Support With VisiCalc - Airplane Investment					
1	Capital Investment	--0--	--0--	Yrs	
2	Interest	0.00	--0--	% APR	
3	Owner Tax bracket	--0--	--0--	%	
4	Owner Hours / week	--0--	--0--		
5	Hourly rental fee	--0--	--0--		
6	Owner's Percentage	--0--	--0--	%	
7	Average fee hrs/mo	--0--	--0--		
8	Fuel cost per gal	--0--	--0--		
9	Gal. Fuel per hr	--0--	--0--		
10	50 hr. check	--0--	--0--		
11	100 hr. check	--0--	--0--		
12	1500 hr. check	--0--	--0--		
13	Insurance / mo	--0--	--0--		
14	Misc. / mo	--0--	--0--		
	/hour	/day	/month	/qtr.	/year

A written code list for template construction can be entered with a minimum amount of confusion if you understand what the code is telling you and how the template is effected by each entry.

Most template codes follow a basic format. Global commands are entered at the beginning. Individual cell information is listed either by rows or by columns — in this case, by columns.

The cells which contain text (labels) are not listed. You can copy them easier from the example template.

Each cell is identified by its coordinate, followed by the formula or format which is to be entered at that location; i.e., "B3:/FL" means put the cursor in column B, row 3, and enter "/FL" for left-justified format.

Using the /R command (replicate) is sometimes easier from a line of coded instructions than from explicit instructions. You don't have to know what it's doing to make it work. Simply type in the exact characters in the code, press return, then enter 'r' and 'n' sequence and watch what happens.

The first few codes are explained in detail as examples of how to trace the formula to the variables they use. The others can be traced easily by comparing their content-variables against the example template above.

The replicate sequence at D23 is a little confusing to some users. Pay close attention to the reverse sequence of the (:) and (.) on the second and third lines. In the first line you set the format to general.

On the second line, with the cursor still at location D23, enter /R for replicate. The edit line will begin with D23. Enter a (:), then E23, then a (.), and H23. Press RETURN. You will not be asked to enter 'r' or 'n' because you are simply replicating the general format from the single cell to each of the cells in the 23rd row from column E through column H.

On the third line, with the cursor still at location D23, enter /R for replicate. The edit line will begin with D23 as it did before. This time enter a (.) instead of a colon, then enter H23, followed by a colon (:), then D24. Press return. In this case, you will copy the series of cells from row 23 to row 24. You won't need to specify "where" on row 24 to end the sequence. That is determined by the "begin" and "end" points specified in the source. The destination is simply row 24 beginning at column D.

/GC9

/GFS

Set Global column width to 9 (the default column width), and Global format to dollars & cents (rounded to 2 digits right of decimal).

B3:1

B3:/FL

B4:/FL

Enter the value (1) in cell B3. Enter left-justified format in cells B3 and B4.

B4:1+B3

B4:/R:B5.B16 r

Enter the formula (1+B3) in cell B4, then replicate the formula from cells B5 to B16 as relative.

D21:((E3/F3)/12)/(((E6*4.34)/30)+E9)

D21:/R:D22 rnnn (note 'r n n n' sequence)

Enter the formula as shown in cell D21 - ((capital investment/years) /12 months) / (((owner hours * 4.34 weeks per month) / 30 days per month) + fee hours per month). Replicate this formula into cell D22 as relative (interest), no change (years), no change (owner hours), no change (fee hours).

D23:/FG

D23:/R:E23.H23 REM /R from cell to line

D23:/R.H23:D24 REM /R from line to cell

Fix the format in cell D23 as (general). Then replicate the format across the row from E23 to H23. Then replicate that sequence on row 23 to begin at cell D24.

D25:+E11*E10

D26:.5*(E12/50)

D27:((E13/100)/18)*17

D28:(E14/1800)

D29:+E15/E9+(E6*4.34))

D30:+E16/E9

D32:(38/100)*E7

D33:(D24*E7)-(D32*D24)

D34:((D32-@SUM(D25.D30)+(.2*D21)*E5/100))

/R D34:E34.H34 rrrrn

D36:@SUM(D32.D34)

/R D36:E36.H36 rr

D37:@SUM(D21.D30)-D23-D24

/R D37:E37.H37 rrrrr

D39:(D36-D37)/R D39:E39.H39 rrrrr

D41:/FG

E4:((E3*((F4/100)*F3))/2

E5:/FG

E6:/FG

E8:/FG

E9:/FG

E11:/FG

E21:((E3/F3)/12)/30

/R E21:E22 rn

E23:+E9/30

E24:((E6*4.34)/30)

E25:D25*(E23+E24)

/R E25:E26.E28 rnn

E29:+E15/30

E30:+E16/30

E32:+D32*(E23+E24)

E33:(E24*E7)-(D32*D24)

F3:/FG

F4:/FG

F21:30*E21

/R F21:F22.F33 r
 G21:3*F21
 /RG21:G22.G33 r
 H21:12*F21
 /R H21:H22.H33 r
 H41:/FI
 H41:@IF(H39<0,F3*@ABS(H39),F3*(-1*H39))

The formula at H41 isn't as complicated as it looks if you break it down into logical steps: If H39 is less than 0..

@IF(H39<0
 (then use) the product of F3 times the absolute value of H39
 ,F3*ABS(H39)
 (else use) the product of F3 times H39 as a negative value
 ,F3*(-1*H39))

which produces a positive number if the bottom line figure is a PROFIT, or a negative number if the bottom line figure is a COST.

USING THE TEMPLATE FOR DECISION SUPPORT

After the blank template has been saved on disk, you're ready to start using it. We can begin with the numbers supplied by the flight service, and our weekend pilot's estimate of tax bracket and weekly flying plans.

\$18,000 Capital Investment figure

- 5 Years
- 14.5 % APR
- 32 % tax bracket
- 3 Owner hours / week
- 37.00 Rental Fee per hour
- 80 % Owner's income from Rental
- 30 Average Fee hours / month
- 1.85 Fuel cost per gal.
- 6.5 Average fuel per hr.
- 150.00 50 hr. maintenance cost
- 300.00 100 hr. maintenance cost
- 5000.00 1800 hr. maintenance cost
- 290.00 monthly insurance premium
- 25.00 Miscellaneous

Looking over the numbers gives our weekend pilot a pretty good picture of what the airplane purchase is going to COST in real dollars. Less than \$6,000 over a five-year period is a more than reasonable price for the equipment. And dropping his rent-to-fly outlay by \$7 an hour isn't too hard to take. The problem is that he's not really in the market to buy an airplane and these numbers make it clear that this isn't a money-making venture.

Our weekend pilot mentions the deal to a friend (another weekend pilot) who, along with being in a higher tax bracket, also flies twice as many hours per week on the average. Plugging his numbers into the template gives an altogether different outcome. We change the tax bracket figure to 42%, the pilot hours to 6 per week, and enter a 14% interest rate (weekend pilot #2's personal banker is more flexible than a finance company).

Based in part on the information derived from the template, Weekend Pilot #2 extracts a 35 fee-hours per month guarantee from the flight service and closes the deal.² Try changing a few more of the variables to determine the investment feasibility. For instance, what if the price of fuel rose to \$2 a gallon? What if the fee-hours increased to 40 per month? Or dropped to 20 per month? What if the owner didn't fly for the

whole month? What if the rental fee was increased to \$38 per hour? Or dropped to \$32?

Using a VisiCalc template for decision support allows you to know the outcome under enough different circumstances to make your decision what they should be. Informed decisions!

Continued from page 37.

```
KIP[22]:= 164; KIP[23]:= 67; { LDY 43 }
KIP[24]:= 96; { RTS }
KIP[25]:= 152; { TYA }
KIP[26]:= 201; KIP[27]:= 193; { CMP #0C1 }
KIP[28]:= 48; KIP[29]:= 246; { BM1 -10 }
KIP[30]:= 201; KIP[31]:= 219; { CMP #0DB }
KIP[32]:= 16; KIP[33]:= 242; { BPL -14 }
KIP[34]:= 138; { TXA }
KIP[35]:= 41; KIP[36]:= 2; { AND #02 }
KIP[37]:= 170; { TAX }
KIP[38]:= 152; { TYA }
KIP[39]:= 224; KIP[40]:= 0; { CPX #00 }
KIP[41]:= 208; KIP[42]:= 233; { BNE -17 }
KIP[43]:= 24; { CLC }
KIP[44]:= 105; KIP[45]:= 32; { ADC #20 }
KIP[46]:= 208; KIP[47]:= 228; { BNE -28 }
KIP[48]:= 0; { BRK }

END; { Of PROCEDURE Set_Keyin_Patch }
```

```
PROCEDURE Make_Patches (OFFSET: INTEGER);
VAR INDEX: INTEGER;
```

```
BEGIN
  MOVELEFT (KIP, T_BUF [OFFSET+510], 49);
  T_BUF [OFFSET+18]:= 127;
  T_BUF [OFFSET+19]:= 234;
  T_BUF [OFFSET+20]:= 234;
  T_BUF [OFFSET+40]:= 74;
  T_BUF [OFFSET+41]:= 254;
  T_BUF [OFFSET+42]:= 254;
  T_BUF [OFFSET+130]:= 234;
  T_BUF [OFFSET+131]:= 234;
  T_BUF [OFFSET+385]:= 127;
END; { Of PROCEDURE Make_Patches }
```

```
BEGIN { Of Patch_Monitor }
  Set_Keyin_Patch;
  Read_Track (5);
  Make_Patches (2048);
  Write_Track (5);
  Read_Track (9);
  Make_Patches (768);
  Write_Track (9);
END; { Of PROCEDURE Patch_Monitor }
```

```
PROCEDURE Patch_Applesoft;
BEGIN
  Read_Track (7);
  T_BUF [354]:= 234;
  T_BUF [355]:= 234;
  Write_Track (7);
  Read_Track (8);
  T_BUF [2168]:= 127;
  T_BUF [2173]:= 234;
  T_BUF [2174]:= 234;
  Write_Track (8);
END; { Of PROCEDURE Patch_Applesoft }
```

```
BEGIN { Of Patch_Character_Set }
  Get_CSet;
  WHILE NOT Emulation_Disk_Verified DO
    BEGIN
      WRITE ('Please replace emulation disk in Unit #',EMUL_UNIT,
            ', and press <RETURN>');
      READLN;
    END;
  Read_Track (0);
  MOVELEFT (C_SET, T_BUF [2694], SIZEOF (C_SET));
  Write_Track (0);
  Patch_Monitor;
  Patch_Applesoft;
END; { Of PROCEDURE Patch_Character_Set }
```

```
PROCEDURE Introduction;
VAR ACK: Char;
```

```
BEGIN
  WRITE (CHR (28)); { Clear screen }
  GOTOXY (2,0); WRITE (CHR (2)); { Set viewport }
  WRITELN ('                                     Apple ][ Emulation Patcher');
  WRITELN ('                                     by Al Evans');
  WRITELN ('   with thanks to George Detzel for the patch routines');
  WRITELN ; WRITELN ;
  WRITELN ('This program will alter an APPLE ][ EMULATION diskette to permit');
  WRITELN ('the use of the upper and lower case characters from any Apple //');
  WRITELN ('character set.');
  WRITELN ; WRITELN ;
  WRITELN ('Before you use the program, you must know two things:');
  WRITELN (' 1. The PASCAL UNIT NUMBER of the disk drive which will hold');
  WRITELN ('     the APPLE ][ EMULATION disk to be modified, and');
  WRITELN (' 2. The PATHNAME of the file containing the character set you');
  WRITELN ('     wish to use in Apple ][ emulation mode.');
  WRITELN ; WRITELN ;
  WRITE ('Press <ESCAPE> if you''re not ready yet, <SPACEBAR> to continue:');
```

Program listing continued on page 20

The Apple Writer WHAT?

by Sharon Webb

In the market for a database? Maybe you don't need to buy one after all. "Uh, oh," I can hear you from here: "What do you know about my needs?", you say. "Look at this stack of paper on my desk. Without a database, how can I get it organized?"

OK. I hear you. But maybe—just maybe—you don't. My husband and I found we could do without one quite nicely. Bryan is in the real estate business. He has to struggle with loads of listings that need constant updating. I'm a writer. Besides articles like this, I write science fiction, and I learned early on that a fiction writer's best friend is his notebook.

Writer's notebooks grow faster than problems on a soap opera. Pretty soon they overflow in stacks and heaps on the table and spill over onto the floor. Transcribing the notes onto a disk isn't the answer either; true, it eliminates the paper, but now stacks of floppies take up the slack. And somewhere in one of them is that absolutely fascinating idea about artificial intelligence I came up with a month or so ago. The question is, "where?"

Obviously, we both felt the need for a database. But, good databases aren't cheap, are they? That's when I took a long and speculative look at Apple Writer ///.

"A word processor?" you say. "C'mon."

Wait. Does Apple Writer /// know it's a word processor? What if we told it to act like a database?

Apple Writer /// has some powerful features not found in most other word processors. One of them allows a single file to be as large as a disk. Another is its sophisticated loading ability. The two work together. If a file can be larger than memory, then it's necessary to be able to load just a portion of that file at a time. That's the secret to fooling Apple Writer /// into acting like a database.

Take real estate for instance: In comes a customer with definite ideas. He wants a three-bedroom, two-bath ranch style house. Throw away those thick listing notebooks. Instead, let's create an Apple Writer /// database.

Boot up Apple Writer ///. Then insert a blank, formatted data disk. At the top of the screen, type this:

CODE:

Below that, type this:

FIELD 1 <25-50k><50-75k><75-100k><100-150k><150-200k><200k+>

Field 1 is the listing's price range, "k" of course, standing for thousand. The figures can be adjusted any way you would like. FIELD 2 holds the number of bedrooms and baths:

FIELD 2 <2,1><2,2><3,2><4,2><5+,2+>

Now we need a search field that will look for the style of the house. If we use "CO" for colonial, "RA" for ranch, "RS" for rustic, "TR" for traditional, and "MO" for modern, Field 3 looks like this:

FIELD 3 <CO><RA><RS><TR><MO>

How about the size of the property? Is this house on a lot? A half-acre? An acre? Field 4 takes care of that:

FIELD 4 <LT><1/2A><A><A+>

Does the house have a pool? Is it a waterfront listing? How about a tennis court? Does it have a fence? Is the property wooded? Or maybe this particular listing is pretty boring with no special features at all. Field 5 might look like this:

FIELD 5 <PL><WF><TN><FN><WD><NO>

Let's add some notes. This will help you remember what those letters stand for later on.

PL=pool

TN=tennis

WD=wooded

WF=waterfront

FN=fence

NO=no special features

Put in other fields if you like—perhaps one to differentiate brick, wood, stucco, CBS construction, or one to designate different neighborhoods. Just don't go overboard, or you'll find your database growing too complicated for easy use.

At the end of the Final field, press RETURN and enter an asterisk:

*[RETURN]

Start typing in those listings now. At the beginning of each, put in the appropriate field information. A three-bedroom, two-bath, ranch-style listing might look like this:

<50-100k><3,2><RA><A+><WD>

Three bedroom, two bath Ranch in Oakmont Hill, Needs work. (And so on...)

*[RETURN]

The descriptive string of characters that begins the listing is your first delimiter. Notice the asterisk at the end of the listing. This is your final delimiter; it prevents those portions of your file that you don't want, from being loaded.

Save the file. Let's call it LISTINGS. To see how your database works, erase memory and type this:

[L].d1/LISTINGS!CODE:!*!

Let's pretend that the disk is full of listings. Your customer is looking for a house on several acres and he insists on woods. Price to him is no object, nor is size.

Let's look at the code: Field 4 has the entry <A+> (for more than one acre). Field 5 holds the code: <WD> (for wooded). Place the cursor at the end of the document in memory (Ctrl E), then type this:

[L].d1/LISTINGS!<A+><WD>!*!A

The final "A" tells Apple Writer /// to load All occurrences of <A+> and <WD>. Make sure that "A" is typed in upper case. Also, be certain that your [L]oad instructions agree with your code. If your listing shows <WD> and you request <wd>, the program will not find a match.

When you press RETURN, every listing on the disk that's coded "multiple acreage" <A+> and wooded <WD> will load into memory beginning at the position of the cursor. If you like, you can load the listings without the delimiters by changing the final "A" to "NA." If, however, you are not requesting a search of all fields (as in the above example), the delimiters not searched for will be loaded into memory. Voila! A word processor as data base!

And so there we have it—Apple Writer /// as the perfect database. Well, no, not perfect. There are limitations, of course. Apple Writer /// can't skip a field. If you try to load the code for Field 1 and Field 3, but leave out Field 2, the program won't be able to find the "word" you've used as a delimiter. And you won't be able to work math manipulations as you can in many commercial databases. Within its limitations, however, Apple Writer /// can be fooled into becoming a useful database—one with no restrictions as to the size of each entry.

continued on page 31

New Products Received

by Bob Consorti

The products outlined below have been received by **ON THREE** for the purpose of review. Some have been reviewed in the past and many will be reviewed in the future. The products have all been given that **ON THREE** 'stamp of approval'. This is only an indication that the product works as advertised and is not an endorsement of the product by **ON THREE**.

/// E-Z Pieces

The advertisements say, "The first fully integrated word processor, spreadsheet and data base manager for the **Apple** ///. Combines the power of the three most commonly used program types with the ability to easily transfer data among them." - I say, "It Works!"

An in-depth review will be in a future issue, but for now let me give my initial impressions of this product. The three sections of the program /// **E-Z Pieces** are quality products in their own right, together they form a very useful package for the **Apple** /// user.

The program author, **Rupert Lissner** is the person who created **QuickFile**. The data base of /// E-Z Pieces appropriately looks very much like QuickFile. With expanded functions and speed, it's a good choice for your small to medium sized information handling needs.

The spread sheet is also very good. Functionally speaking, it works much faster than either version of VisiCalc for the /// but it doesn't have all the features of Advanced VisiCalc. It will directly read VisiCalc and DIF files at very quick speeds, so don't be surprised when that large spread sheet that takes a half an hour to load under VisiCalc, loads in a minute under /// E-Z Pieces.

The best part of the package is the word processor. Much better than Apple Writer, it comes very close to being better than Word Juggler. In a head on head comparison Word Juggler would win, yet because /// E-Z Pieces is an integrated package it wins hands down.

This "Integration" allows you to directly cut a portion of your data base or spread sheet and transfer it directly to the word processor. Up to 12 files may be resident on the /// E-Z Pieces "Desktop" as you use the program and you can quickly move from one to another.

While it is a very good package it does have a number of small bugs that will be touched on in the future. This is to be expected with a new program and I'm sure that they will be ironed out in the future. The only major disappointment is the manual. A few hundred pages long it looks as if it was written as an afterthought. It's a concise reference manual that needs some work. It's possible that a single book can't fully describe this type of program and I think they should have split it up.

All in all a very nice package, that works nicely. Available from a limited number of dealers, the package is priced at \$295 and will run on a 128K or 256K **Apple** /// system.

haba systems, inc., 15154 Stagg Street, Van Nuys, California 91405. (213) 901-8828.

Catalyst Version 2.0

The best selling hard disk boot program for the **Apple** /// has been improved with many of the problems of Version 1.0 fixed. For those of you not familiar with **Catalyst**, it will install

most any **Apple** /// program on your Apple ProFile hard disk. You can then choose the program from a menu and not have to restart the entire computer system each time you wanted to change a program.

The most interesting feature of Catalyst Version 2.0 is the automatic program installation feature. Just insert the right diskettes and Catalyst will automatically put the programs on your hard disk.

There is also a new ability to dynamically load drivers. All the drivers that your programs need do not have to be on your startup disk. For example, you can have Catalyst load in the .GRAFIX driver only when needed by a program such as Business Graphics. This leaves more space for your other programs.

However, even with dynamic loading of drivers, Catalyst 2.0 takes up about the same amount of memory as the original Catalyst. This is due to the new enhancements. The current procedure for installing Pascal programs still has some major problems. For example, if you have a copy of our **Draw ON** graphics tool and the System Utilities or Lazarus installed under Catalyst there are problems.

Draw ON /// requires that 32K of memory be reserved for graphics. If you do this you will not be able to directly use the Utilities program or **Lazarus** since they both require that 0K of memory be reserved for graphics. To access these programs you will have to manually change the graphics space allocation before trying to use them. This is a very annoying problem that should be fixed.

A sparse, non-typeset user's manual is written on a technical level that is above most of the people who use the ///. For the \$149 price tag it should be a bit better. While it will still work with only the ProFile and Micro-Sci drives it is an excellent package that will improve your productivity. Quark Incorporated, 2525 West Evans, Suite 220, Denver, Colorado 80219. (303) 934-2211.

KEYSTROKE Data Base & Report Generator

The **KEYSTROKE Data Base & Report Generator** are the first in a series of **Apple** /// business software that Brock Software Products, Inc. has announced. Very attractively packaged, these products look great from the outside.

From the inside however there are some problems. While offering very powerful options and real relational features, the program is set up on a less than logical menu structure. Based on our tests, users have a hard time getting to learn the program but with careful study of the excellent manual they can master it.

The **KEYSTROKE** data base allows you to design your input forms just like PFS. Defaults, computed fields and automatic field formatting make data entry a very easy process. Capabilities exist for merging VisiCalc, Apple Writer and PFS files yet it is not integrated in the respect that the program can't directly read those other types of files, a conversion process is needed.

If you can master it, this can be a very valuable addition to your **Apple** /// program library. The relational features are really great. Cross referencing information from a number of files is a very important feature that is not found in most data base packages today.

Brock Software Products, Inc., 8603 Pyott Road, Crystal Lake, Illinois 60014. (815) 459-4210.

SOSTRAN, FONTWRITER, EASYTERM ///, INFONET

With the introduction of these new products for the /// Sun Data, Inc. brings **Apple** /// users a number of low cost high quality items.

SOSTRAN is a program that copies files from your **Apple** /// format disks to your **Apple II** disks and vice-versa. This very useful utility allows you to transfer ANY type of file from one machine to the other. Basic, programs, text and data files, DIF Visicalc files, and even graphic photos can be transferred. The only unusual thing about this product is the manual - Typeset and bound in a binder, it's too good for the price!

FONTWRITER is a set of three powerful and easy to use utility programs for the ///. With FONTWRITER you can easily 1) Create and modify character fonts for the text and graphic screens. 2) Create fonts to form large logo-type displays and produce animated characters on the screen. 3) Modify the keyboard layout providing the user with complete control over what happens when a key is pressed.

Again the strongest point of the package is the excellent documentation. The utility programs are very well described and are very easy to use. Complete with Appendices and tutorials, the manual makes operation of the program a snap.

EASYTERM /// is a communications program that gives your computer the ability to send files created on your **Apple** /// to a remote computer and to receive data from a remote computer. Once again the manuals are excellent, better than most of the more expensive packages.

The EASYTERM program is written completely in assembly language as an interpreter. This results in lightning fast speeds. It doesn't have the ability to autodial numbers or some of the other things the more expensive packages offer but it is a great buy for the money. You can change your communications protocols and the general status of the program directly within the program. Everything occurs very quickly.

INFONET is an information network program for those who need electronic communications between themselves and other computer users. It allows your **Apple** /// to act as an electronic bulletin board / message center. The users need only have a computer with a modem.

Assigned users are allowed to communicate with INFONET at different levels, depending on the priority level assigned to them. INFONET can accommodate over 1,000 users, depending on the size of the hard disk used.

Another package with an excellent owner's manual, **ON THREE** will be using this system when we set up our long-awaited bulletin board in a month or so.

All of these products can be ordered from Sun Data, Inc., 95 West 100 South, Logan, Utah 84321. (801) 752-7831.

Basic Extension

Basic Extension is a group of routines for the **Apple** /// Business Basic programmer that act as an extension to the Basic language. Written in assembly language, most of these routines will speed development work and program speed of Basic programs.

There are three general routines available: Disk, Array, and Utility. These are briefly documented in a sparse 5 page manual. While sparsely documented on paper, there are demonstration programs that show you how to use all the functions of the new routines.

The disk routines provide high speed disk read/write capabilities. You can speed up disk access by as much as ten times over the standard **Apple** /// Business Basic routines. A direct block-access program gives you access to any block of data on your diskettes. Also included in the disk routines is the ability to change a file type from within Basic.

The array routines provide high speed manipulation of numeric and string arrays. Insert and delete elements, move sections of an array and even transferring information from one array to another are all now possible. You can also search entire arrays for an occurrence of a single string.

In comparison to the fast disk I/O and the lightning fast array searches, the utility routines are almost primitive. Many of these utilities (or ones very similar) have been published in **ON THREE** magazine. The Font editor and Block editor supplied as demonstration programs are among the slowest and least sophisticated I've seen.

The power of this package lies not in the demonstration programs (as the name implies) but with the assembly language extensions to Business Basic that can speed things up tremendously. Foxware Products will allow your programs to use these routines for a small fee. It's not the most powerful utility package for the /// but it does have a very useful purpose.

Foxware Products, 2506 W. Midwest Drive, Taylorsville, Utah 84118. (801) 364-0394.

Attach Driver

The **Attach Driver** is an **Apple** /// SOS Driver that allows text screen dumps from within any program. Compatible with Catalyst, this Driver enables text screen dumps of both 40 and 80 columns.

This Driver doesn't allow printouts of graphic screens - only text screens. It will however allow two special codes to be sent to your printer before each print. You can use these codes to toggle from two of your printer modes such as normal and condensed print.

One of the "Features" of this driver is that a character will appear in the upper right hand corner of the screen before it prints. This is very annoying as it results in your screen being changed. On your printouts this character will also appear.

It does work in all the programs that we have tested. In addition to the small problem reported above, the documentation is somewhat lacking. This doesn't really create a problem as enough is presented to get the driver onto your system, but a typeset instruction sheet would be nice. The Attach Driver lists for \$29.95.

SOFT-LIFE CORPORATION, 2950 Los Feliz Blvd. Suite 103, Los Angeles, California 93039.

Aladin

Aladin is a very powerful relational data base and problem solver. Very sophisticated, it brings relational data base and mainframe power to the **Apple** ///. Direct sorting capabilities, unlimited "key" fields and over 1 million records per file make this a very attractive data base system for the **Apple** ///.

Talk about powerful, this package defies review. I should start out with what it will not do: 1) It won't walk your dog; 2) It won't cut your hair. Aside from that I'm not sure what it can't do. Almost everything that you could want in a data base is here.

Fully relational, Aladin can access any number of files at once to give you the information you request. Report generation is built-in with sophisticated options for print formatting. A full

Query language is also implemented. For those of you who aren't data processing professionals, this simply means that you can access your data base with English like commands - not numbers or complicated formulas.

There is a built-in provision for integrating with a stand-alone word processor such as Apple Writer. If you like the WPL of Apple Writer, wait until you see what this package can do! It's been said the WPL is a mini programming language inside the word processor. If that's true (and it is) then Aladin has a built-in language that looks like it came from a mainframe - it's powerful.

Statistical analysis and charting also are built-in. I don't know any other words to describe this package but - **POWERFUL**. For more information contact: The Advanced Data Institute, America Inc., New Products Division, 1215 Howe Avenue, Sacramento, California 95825.

Copy //

Copy // is a diskette copying utility program for the **Apple //**. Simply put, it will do a complete diskette copy and verification. Reading an entire 140K floppy disk into memory, it will make a copy of the disk in memory and then check the copied disk to make sure that there are no errors.

Copy // uses a very accurate verification routine to make sure that you have created an exact duplicate of your original diskette. Extensive options allow you to format the diskettes to

be copied and copy up to four disks sequentially at a time.

Copy // will also do disk copies with just the internal disk drive. Because the program reads the entire diskette into memory, a 256K **Apple //** is needed. For those of you who need accurate verification that the copies you create are exact duplicates, Copy // is for you. Please note that it will only work on standard 140K **Apple //** disk drives, the Micro-Sci's and others are not supported.

Digital Microware, P.O. Box 289 Los Olivos, California 93441. (714) 855-0555.

Timesaver Taxes

Timesaver Taxes is a tool for the professional income tax preparer. It will print all the common IRS forms and permit the totals from various other forms to be entered and used in the tax calculations.

This program provides a fast and easy-to-use approach to individual income taxes. An aid to the professional preparer, it is not a substitute for the preparer's judgment. The capabilities of the **Apple //** are used to speed calculations and make data entry very easy.

Annual updates are available that reflect the changes in the tax laws. State income tax packages are available upon request. C & H TAXES, Suite 29, 2002 So. St. Aubin, Sioux City, Iowa 51106. (712) 276-3362.

WHAT YOU SEE IS WHAT YOU GET (on your screen) (on your printer)

- VisiCalc printouts in condensed print - easily;
- Print a catalog of Apple Writer files;
- Print Help screens from any program;

**ASK FOR
ATTACH DRIVER.**

A1	(L)	SOFT-LIFE'S NEW 'ATTACH.DRIVER' PRINTS YOUR SCREEN!	CP					
			161					
1	A	B	C	D	E	F	G	H
2	LOOK!! ->-> APPLE // TEXT SCREEN DUMP							
3								
4	Look at this!!! A screen dump from an Apple // by							
5	using the new "Attach.Driver". NO EXTRA HARD-WARE!!							
6	WORKS WITH ANY PRINTER AND WITH ANY S.O.S. PROGRAM!!							
7	You can make copies of your menus or your help screens							
8	or a page full of any text whatsoever!! Works with							
9	Pascal and with Basic. Works from within a commercial							
10	program or from the programs you write for yourself.							
11								
12	NOTE: THIS IS A TEXT SCREEN DUMP DRIVER AND DOES NOT							
13	PRINT GRAPHICS.							
14	BONUS: NOW you can ALSO send any two printer codes to your							
15	printer. This means you could toggle your dot-matrix							
16	printer from condensed to normal and back to condensed							
17	with a single keystroke!! Or pick any two functions of							
18	your particular printer and they can be toggled on and							
19	off without fuss or muss!!							
20								

- Make hard copy printouts of menus;
- Works with any printer - any printer driver;
- Works with any SOS program.

Only \$29.95 + tax
(add \$1.50 for shipping)

"Actually printed from a VisiCalc program"

New Products Coming! Exclusively for
Apple III! Write for Details.



SOFT-LIFE CORPORATION
2950 LOS FELIZ BLVD., SUITE 103,
LOS ANGELES, CALIFORNIA 90039

Book Review

by Dana E. Wilson, M.D.

ELECTRONIC LIFE. How to Think About Computers. Michael Crichton. Alfred A. Knopf. New York, 1983. 209 pp. 12.95 retail.

The author of **The Andromeda Strain, Five Patients and Eaters of the Dead** has turned his considerable talents loose on a book about computers! Lest you wonder whether a physician, writer and film maker is a dabbler and Johnny-Come-Lately to the computer scene, Dr. Crichton has been involved with computers since the mid-60's and pioneered in introducing computer graphics to the film industry. He is a highly informed aficionado who transmits his own enthusiasm to his fans.

Crichton began the book as a series of instructions to friends, written on his word processor. These instructions grew to monograph size. The book might be mistaken for a catalog of computer terms. Don't dismiss it as another "ABC" computer glossary, though! Crichton subtitled it "How to Think About Computers" and therein lies the key to its appeal. It is an effective exposition of the author's own perspectives about computers, even though its informational content is slim. Each of the entries is really a micro-essay filled with humor, observations, and philosophy. As a glossary it's inadequate, and because of its glossary format, it is sometimes repetitive. But as an example of good writing, it's a gem. It is pitched at the beginner, but there is something in it for everyone.

Several **BASIC** programs are appended. They are intended to introduce beginners to computers, and give them the confidence needed to progress further. This will be of little interest to **ON THREE'S** readership.

Electronic Life more than vaguely resembles Christopher Evan's *The Making of the Micro Millennium*. If you liked Evan's you will enjoy Crichton, repetition notwithstanding. His concise sentence structure reflects careful thought, like an elegant line of code.

Why am I reviewing this book in **ON THREE?** Because Crichton did us a disservice twice! He capitalizes on **Apple** ///'s difficult birth as an example of why one should never buy a micro early in its marketing cycle. Sadly, he never points out that the // has become an outstanding machine. Secondly, he

implies that it is essential to have CP/M. Neither position is fully explained. The // ends up the worse for it.

My recommendation to **ON THREE**'s readers, unless you are anxious to introduce a novice to the mysteries of computing, wait until Electronic Life appears in paperback. The book is worth reading, but for \$12.95, Apple // owners can wait until later in its marketing cycle! //

Call THREE: Hot Line

The **Call THREE: Hot Line** is a service whereby **Apple** // users with problems can call a relatively local number to get help. The people answering the phones are fellow **Apple** // users who have volunteered to help others over the rough spots.

If you know enough about your machine to answer questions, please write in so we can include you in a future listing. For those who write in, please state your areas of expertise and the hours during the week you are willing to take calls.

For those of you who need questions answered: **PLEASE** call only within the hours specified. Also note any time zone differences. If these volunteers start getting calls at all hours, the service will have to be discontinued, so **PLEASE** follow this guideline!

Legend

Accounting: AC	Graphics: GR
Assembly Language: AL	Pascal: PA
Cobol: CO	Spread Sheet: SS
Education: ED	Quark: QU
General Use: GE	Agriculture: AG
Modems: MD	Catalyst: CT
SOS: SO	Data Base: DB
Word Processing: WP	Fortran: FO
J/Emulation: AE	Micro-Sci: MI
Business Basic: BB	Profile: PR
CP/M: CP	Telecom.: TC
Financial: FI	

Under the topics column, there are two character mnemonics that specify the individuals area(s) of expertise. They are outlined above.

Hot Line Listing

Name	State	Phone Number	Days	Hours	Topics
Coville Woodburn	NH	(603) 863-5590	M,Tu,Th,F	7-8PM	CT, QU
Harry T. Hanson, Ph.D.	NJ	(201) 467-0712	M-F	6-9PM	GE
Edward N. Gooding Sr.	VA	(804) 747-8751	M-Su	6-9PM	CO, SS, PR, MD, CT
Al Johnston	FL	(904) 739-1042	M-F	9AM-6PM	GE
Paul Sanchez	FL	(305) 266-5965	M-Su	10AM-4PM	SS, PR, CT
John & Lisa Beckett	MO	(417) 678-2500	M-F	6-9PM	GE
David B. Hays	KS	(316) 722-1242	M-F	7-11PM	GE
Art Schumer	ND	(701) 282-7907	M-F	6-10PM	AL, BB, GR, SO, SS, AE
Terri Wiles	CO	(303) 850-7472	M-Su	10AM-6PM	PA
Jeff Fritz	CA	(415) 864-2600	M-F	6-9PM	BB
Carl & Anita Reynolds	CA	(714) 734-9324	M,Tu,F	4PM-9PM	GE
Wayne Hale	CA	(619) 450-3856	M-F	7-11AM	BB, GR, CT
Dennis R. Cohen	CA	(213) 956-8559	Su-Th	10AM-10PM	GE
Kelly C. McGrew	WA	(206) 943-8533	M-Su	6-10 PM	DB, GR, PA, SS, PR, MD, CT

Continued from page 48.

```

PROGRAM Radiate;
{ *****
  * Radiate -- by Dennis Cohen
  * -----
  * This menu driven program will present you with three
  * colorful graphic demonstrations.
  ***** }

USES PGRAF, REALMODES, TRANSCEND, APPLESTUFF;

CONST
  xmax = 139 { Set it up for COL140 };
  ymax = 191;

VAR
  number: integer;

FUNCTION rndcolor: screencolor { Utility function };
  VAR
    color: screencolor;
  BEGIN
    CASE (1 + (random MOD 16)) OF
      1: color := black;
      2: color := magenta;
      3: color := darkblue;
      4: color := purple;
      5: color := darkgreen;
      6: color := grey1;
      7: color := mediumblue;
      8: color := lightblue;
      9: color := brown;
      10: color := orange;
      11: color := grey2;
      12: color := pink;
      13: color := green;
      14: color := yellow;
      15: color := aqua;
      16: color := white;
    END; { Of CASE }
    rndcolor := color;
  END; { Of FUNCTION rndcolor }

PROCEDURE fourcorners;
  CONST
    maxchange = 100;
  VAR
    angle1, angle2, angle3, angle4: real;
    c1, c2, c3, c4, d1, d2, d3, d4: integer;
  PROCEDURE line (x1, y1: integer; var al: real; d: integer);
    VAR
      x2, y2: integer;
    BEGIN
      pencolor (rndcolor);
      moveto (x1, y1);
      x2 := x1 + TRUNC (ymax * COS (al));
      y2 := y1 + TRUNC (ymax * SIN (al));
      lineto (x2, y2);
      al := al + d;
    END; { of PROCEDURE line }

    BEGIN { Of PROCEDURE fourcorners }
      initgrafix;
      angle1 := 90;           d1 := -1;           c1 := -1;
      angle2 := 270;          d2 := 1;            c2 := 1;
      angle3 := 270;          d3 := -1;          c3 := -1;
      angle4 := 90;           d4 := 1;            c4 := 1;
      grafmode (COL140, 1);
      fillcolor (rndcolor);
      fillport;
      graficon;
      WHILE (d1) > maxchange DO
        BEGIN
          WHILE (angle1) >= 0 DO
            BEGIN
              IF keypress THEN
                BEGIN
                  UNITCLEAR ();
                  EXIT (fourcorners);
                END;
              line (0, 0, angle1, d1);
              line (0, max, angle2, d2);
              line (max, max, angle3, d3);
              line (max, 0, angle4, d4);
            END;
          d1 := d1 + c1; d2 := d2 + c2; d3 := d3 + c3; d4 := d4 + c4;
        WHILE (angle1) <= 90 DO
          BEGIN
            IF keypress THEN
              BEGIN
                UNITCLEAR ();
                EXIT (fourcorners);
              END;
            line (0, 0, angle1, -d1);
            line (0, max, angle2, -d2);
            line (max, max, angle3, -d3);
            line (max, 0, angle4, -d4);
          END;
        d1 := d1 + c1; d2 := d2 + c2; d3 := d3 + c3; d4 := d4 + c4
      END; { Of PROCEDURE fourcorners }

      PROCEDURE screenfill;
      CONST
        maxchange = 100;
        xcenter = 70;
        ycenter = 96;
      VAR
        change, x, y: integer;
      PROCEDURE line (x1, y1, x2, y2: integer);
        BEGIN
          pencolor (rndcolor);
          moveto (x1, y1);
          lineto (x2, y2);
        END; { Of PROCEDURE line }

        BEGIN { Of PROCEDURE screenfill }
          initgrafix;
          x := xcenter; y := 0;
          fillcolor (rndcolor);
          fillport;
          grafimode (COL140, 1);
          graficon;
          FOR change := 1 TO maxchange DO
            BEGIN
              WHILE (y) <= max DO
                BEGIN
                  IF keypress THEN
                    BEGIN
                      UNITCLEAR ();
                      EXIT (screenfill);
                    END;
                  line (xcenter, ycenter, x, y);
                  y := y + change;
                END;
              WHILE (x) <= max DO
                BEGIN
                  IF keypress THEN
                    BEGIN
                      UNITCLEAR ();
                      EXIT (screenfill);
                    END;
                  line (xcenter, ycenter, x, y);
                  x := x + change;
                END;
              WHILE (y) >= 0 DO
                BEGIN
                  IF keypress THEN
                    BEGIN
                      UNITCLEAR ();
                      EXIT (screenfill);
                    END;
                  line (xcenter, ycenter, x, y);
                  y := y - change;
                END;
              WHILE (x) >= 0 DO
                BEGIN
                  IF keypress THEN
                    BEGIN
                      UNITCLEAR ();
                      EXIT (screenfill);
                    END;
                  line (xcenter, ycenter, x, y);
                  x := x - change;
                END;
            END; { Of PROCEDURE screenfill }

            PROCEDURE circleradiate;
            CONST
              maxchange = 12;
              maxradius = 60;
            VAR
              xcenter, ycenter, radius: integer;
            PROCEDURE circle (xc, yc, ra: integer);
              CONST
                maxrotations = 4;
              VAR
                angle: real;
                change: integer;
                color: screencolor;
              BEGIN { Of PROCEDURE circle }
                FOR change := 1 TO 1 + (random mod maxrotations) DO
                  BEGIN
                    angle := 0;
                    WHILE (angle) <= 360 DO
                      BEGIN
                        IF keypress THEN
                          BEGIN
                            UNITCLEAR ();
                            EXIT (circleradiate);
                          END;
                        line (xc + TRUNC (0.56 * ra * COS (angle)),
                              yc + TRUNC (ra * SIN (angle)));
                        angle := angle + change;
                      END;
                  END; { Of PROCEDURE circle }

                  BEGIN { Of PROCEDURE circleradiate }
                    initgrafix;
                    grafimode (COL140, 1);
                    fillcolor (rndcolor);
                    fillport;
                    graficon;
                    WHILE NOT keypress DO
                      BEGIN
                        xcenter := random MOD (xmax + 1);
                        ycenter := random MOD (ymax + 1);
                        radius := (random MOD maxradius) + 10
                          { make sure it's at least 10 dots };
                        circle (xcenter, ycenter, radius)
                      END;
                  END; { Of PROCEDURE circleradiate }

                  BEGIN { Of MAIN Program }
                    REPEAT
                      WRITE (chr(20));

```

ON THREE

```

GOTOXY (40, 5); WRITE ('RADIADE');
GOTOXY (20, 9); WRITE ('0 -- Quit');
GOTOXY (20, 9); WRITE ('1 -- Simple Screen Radiale');
GOTOXY (20, 10); WRITE ('2 -- Circle Radiale');
GOTOXY (20, 11); WRITE ('3 -- Radiale from 4 Corners');
GOTOXY (0, 23); WRITE ('Which? ');
texton (Now that the screen's full, turn it on );
READLN (number) (Non-numeric input will crash the program );
CASE number OF
 1: screenfull;
 2: circleradiale;
 3: fourcorners;
END;
UNTIL (number = 0)
END; ( Of PROGRAM Radiale )



---


PROGRAM Beatles;
{
 **** Beatles -- by Dennis Cohen ****
 **** -----
 **** This program will sound out some of the favorite ****
 **** Beatles' melodies of all time - on your Apple //! ****
 **** -----
}
USES APPLESTUFF;

CONST { Defining the notes and tones for Apple // }
  B0 = 47; A0 = 45; G0 = 43; F0 = 41; E0 = 40; D0 = 39;
  C1 = 48; B1 = 50; E1 = 52; F1 = 53; G1 = 55; A1 = 57;
  B2 = 56; C2 = 58; D2 = 60; E2 = 64; F2 = 65; G2 = 67;
  A2 = 69; B2 = 71; C3 = 72; Sharp=1; Flat=-1; Full = 80;
  Half=40; Eighth=10; Threq=80; Quart=20;
  (IS=1)

PROCEDURE Page3;
BEGIN
  WRITELN (CHR (28));
END;

PROCEDURE Sgt;
BEGIN
  Page3; GOTOXY (0, 10);
  ( bar 1 ) WRITE ('We're'); Note (E, Quart);
  ( bar 2 ) WRITE ('Sgt. Pepper's'); Note (G, Quart);
  Note (G, Quart); Note (G, Quart);
  ( bar 3 ) WRITE ('Lonely Hearts'); Note (C2, Half);
  Note (C2, Quart); Note (B + Flat, Half);
  ( bar 4 thru 6 ) WRITELN ('Club Band'); Note (A, Half); Note (G, Full);
  Note (G, Quart); Note (A, Quart); Note (A, Half);
  Note (A, Quart); Note (G, Quart);
  ( bar 7 and 8 ) Note (G, Full); Note (G, Threq);
  WRITE ('We're'); Note (E, Quart);
  ( bar 2 ) WRITE ('Sgt. Pepper''s');
  Note (G, Quart); Note (G, Quart);
  Note (G, Quart); Note (G, Quart);
  ( bar 3 ) WRITE ('Lonely Hearts'); Note (C2, Half);
  Note (C2, Quart); Note (B + Flat, Half);
  ( bar 4 ) WRITELN ('Club Band'); Note (A, Half);
  Note (G, Full); WRITE ('Sit back'); Note (G, Quart);
  ( bar 10 ) Note (A, Quart);
  WRITE LN ('and let the evening go--'); Note (A, Quart);
  Note (A, Quart); Note (A, Quart);
  ( bar 11 ) Note (A, Half);
  Note (A, Quart); Note (G, Threq);
  ( bar 12 thru 14 ) Note (G, Threq); Note (G, Threq);
  WRITE LN ('Sergeant Pepper''s Lonely');
  Note (A, Half); Note (A, Quart);
  Note (A, Quart); Note (A, Quart);
  ( bar 15 thru 20 ) Note (A, Half); Note (A, Quart);
  WRITE LN ('Sergeant Pepper''s Lonely');
  Note (G, Half); Note (G, Quart);
  Note (G, Quart); Note (G, Quart);
  Note (G, Half); Note (G, Quart);
  WRITE LN ('Sergeant Pepper''s Lonely');
  Note (A, Half); Note (A, Quart);
  Note (A, Quart); Note (A, Quart);
  Note (A, Half); Note (A, Quart);
  WRITE LN ('Hearts Club Band'); Note (A, Half);
  Note (A, Half); Note (G, Full);
  END; (sgt)
}

PROCEDURE Ppbck;
BEGIN
  Page3;
  ( bars 1 thru 5 ) WRITELN ('Dear Sir or Madam');
  Note (E, Eighth); Note (F, Eighth);
  Note (G, Quart); Note (G, Quart);
  Note (G, Eighth); Note (G, Eighth);
  WRITE LN ('Will you read my book?'); Note (F, Eighth);
  Note (E, Eighth); Note (G, Quart);
  Note (G, Quart); Note (G, Eighth);
  WRITE LN ('I took me years to write,');
  Note (G, Eighth); Note (G, Eighth);
  Note (A, Eighth); Note (B + Flat, Quart);
  Note (B + Flat, Quart); Note (B + Flat, Quart);
  WRITE LN ('Will you take a look?');
  Note (A, Eighth); Note (G, Eighth);
  Note (A, Eighth); Note (G, Eighth);
  Note (G, Half); Note (E, Eighth);
  Note (E, Eighth); Note (F, Eighth);
  bars 6 thru 10 ) WRITE ('Based on a novel');
  Note (B, Quart); Note (G, Eighth);
  Note (B, Eighth); Note (G, Eighth);
  Note (B, Eighth); WRITE LN ('by a man named Lear');
  Note (F, Eighth); Note (E, Eighth);
  Note (G, Quart); Note (G, Quart);
  Note (G, Quart); Note (B, Quart);
  Note (B, Eighth); Note (A, Eighth);
  Note (B + Flat, Quart); Note (B + Flat, Quart);
  Note (B + Flat, Quart);
  WRITE LN ('I want to be a paperback writer,');
  Note (A, Eighth); Note (G, Eighth);
  Note (A, Eighth); Note (C1, Eighth);
  Note (C1, Eighth); Note (C1, Eighth);
  Note (E, Quart); Note (F, Quart); Note (F, Full);
}

```

(bars 11-13) WRITELN ('PAPERBACK WRITER!');

Note (0, Eighth DIV 2); Note (G, Quart);
Note (G, Eighth); Note (B + Flat, Quart);
Note (C2, Threq); Note (C2, Three);
END; (Ppbck)

PROCEDURE Martha;

BEGIN

Page3; GOTOXY (0, 10);
 (bar 1) WRITELN ('Martha, my dear');
 Note (G, Quart); Note (B + Flat, Half); Note (C2, Quart);
 (bar 2) Note (G, Half); WRITE ('Thought I');
 Note (B + Flat, Quart); Note (C2, Quart);
 (bar 3) WRITE ('I spend my');
 Note (G, Threq); Note (A, Bus+1);
 (bar 4) WRITE LN ('days in conversation');
 Note (B + Flat, Quart); Note (B + Flat, Quart);
 Note (B + Flat, Quart); Note (A, Quart);
 (bar 5) Note (C2, Half); Note (B + Flat, Half);
 (bar 6) WRITE LN ('please'); Note (A, Full + Quart);
 (bar 7) WRITE LN ('Remember me?'); Note (B + Flat, Quart);
 Note (A, Quart); Note (G, Quart);
 (bar 8) Note (F, Full);
 (bar 9 and 10) WRITELN ('Martha, my love');
 Note (A + Flat, Quart); Note (B + Flat, Quart);
 Note (C2, Quart); Note (B + Flat, Full + Quart);
 (bar 11) WRITE LN ('Don't forget me');
 Note (A + Flat, Quart); Note (C2, Quart);
 Note (C2, Quart); Note (D2, Quart + Full);
 (bar 13) WRITE LN ('Martha, my dear!');
 Note (A + Flat, Quart); Note (B + Flat, Quart);
 Note (C2, Quart); Note (B + Flat, Quart); Full);
 END; (martha)

PROCEDURE Fields;

BEGIN

Page3; GOTOXY (0, 10);
 (bar 1) WRITELN ('Let me take you down');
 Note (B, Eighth); Note (B, Eighth);
 Note (C2, Quart); Note (B, Quart);
 (bar 2) Note (G, Quart);
 WRITE LN ('cause I'm goin'' to');
 Note (D, Eighth); Note (E, Eighth);
 Note (A, Quart); Note (G, Quart);
 (bar 3) Note (F, Half); WRITE LN ('Strawberry Fields');
 Note (F, Quart); Note (G, Quart); Note (A, Quart);
 (bar 4) Note (D, Full);
 (bar 5) WRITE LN ('Nothing is real'); Note (0, Quart);
 Note (F, Quart); Note (G, Quart); Note (A, Flat, Quart);
 (bar 6) Note (D, Half + Eighth); Note (D, Quart);
 WRITE LN ('And nothing is real about');
 Note (D, Eighth); Note (C2, Eighth);
 (bar 7) Note (B, Eighth); Note (A, Eighth);
 Note (G, Sharp, Eighth); Note (A, Eighth); Note (B, Quart);
 (bar 8) Note (0, Half);
 WRITE LN ('Strawberry Fields Forever');
 Note (B, Eighth); Note (B, Eighth);
 Note (E, Eighth); Note (B, Eighth);
 Note (G, Eighth); Note (D, Eighth);
 (bar 9) Note (A, Eighth); Note (G, Threq);
 END; (fields)

PROCEDURE Sixtyfour;

BEGIN

Page3; GOTOXY (0, 10);
 (bar 1) WRITELN ('When I get older');
 Note (E, Eighth); Note (Eighth DIV 2);
 Note (D, Sharp, Eighth DIV 2); Note (E, Eighth);
 Note (G, Eighth + Quart); Note (E, Quart);
 (bar 2) WRITE LN ('Losing my hair');
 Note (G, Eighth); Note (Eighth DIV 2); Note (A, Eighth);
 Note (D, Eighth); Note (C2, Eighth + Half);
 (bar 3) WRITE LN ('Many years from now');
 Note (C2, Eighth); Note (E2, Quart + Eighth);
 Note (C2, Quart); Note (A, Quart);
 (bar 4) Note (D2, Quart);
 Note (E, Eighth + (Eighth DIV 2)); Note (A, Eighth DIV 2);
 Note (B, Eighth + (Eighth DIV 2)); Note (A, Eighth DIV 2);
 Note (G, Quart);
 (bar 5) WRITE LN ('Will you still be sending me a valentine?');
 Note (B, Eighth + (Eighth DIV 2)); Note (C2, Eighth DIV 2);
 Note (C2, Sharp, Eighth + (Eighth DIV 2));
 Note (D2, Eighth DIV 2);
 Note (B, Eighth); Note (Eighth DIV 2);
 Note (C2, Sharp, Eighth + (Eighth DIV 2));
 Note (D2, Eighth DIV 2);
 (bar 6) Note (B, Eighth); Note (B + Flat, Quart);
 Note (A, Half + Eighth);
 (bar 7) WRITE LN ('Birthday greetings,');
 Note (C2, Quart); Note (C2, Quart);
 Note (C2, Sharp, Quart); Note (D2, Quart);
 (bar 8) WRITE LN ('battle of wits');
 Note (E2, Eighth);
 Note (E2, Flat, Eighth); Note (D2, Eighth);
 Note (C2, Quart + Eighth); Note (G, Quart);
 (bar 9) WRITE LN ('I'd been out till');
 Note (E, Eighth); Note (Eighth DIV 2); Note (Sharp, Eighth DIV 2);
 Note (E, Eighth); Note (G, Eighth + Quart); Note (E, Quart);
 (bar 10) WRITE LN ('quarter to three');
 Note (G, Eighth); Note (Eighth DIV 2); Note (A, Eighth);
 Note (B, Eighth); Note (C2, Eighth + Half);
 (bar 10) WRITE LN ('Would you lock the door?');
 Note (E2, Eighth); Note (E2, Quart);
 Note (D2, Quart + Eighth); Note (A, Quart);
 (bar 11) Note (C2, Full);
 (bar 12) WRITE LN ('Will you still need me?');
 Note (C2, Eighth); Note (Eighth DIV 2);
 Note (A, Eighth DIV 2); Note (C2, Eighth);
 Note (C2, Flat, Quart); Note (D2, Eighth + Half);
 (bar 13) WRITE LN ('When I'm Sixty Four');
 Note (B, Eighth); Note (Eighth DIV 2);
 Note (B, Quart); Note (A, Quart + Eighth);
 (bar 14) WRITE LN ('When I'm Sixty Four');
 Note (E2, Eighth); Note (E2, Quart + Eighth);
 Note (E2, Quart); Note (E2, Quart);
 (bar 15) Note (C2, Half + Quart); Note (0, Quart);
 END; (sixtyfour)

BEGIN (MAIN Program)

Sgt;
Fields;
Martha;
Sixtyfour;
Ppbck;
Sgt;

END; (Of PROGRAM Beatles)

Three Shorts

(Well, they're not that long!)

by Dennis Cohen

There are three programs listed here, two in **Pascal** and one in **BASIC**. The BASIC program, **CIRCLE**, requires that **/BASIC/BGRAF.INV** be online and is really just a part of the Pascal program **RADIATE**, and generates randomly sized and located circles of multiple, constantly changing colors on the screen.

RADIATE is a menu-driven collection of three graphics demos taken from Tom Swan's UCSD Turtlegraphics-based Moire routines and modified to take advantage of some of the ///'s rather unique capabilities. All three of the demos use the COL140 mode, so that if you want to use CP280 you'll need to change some global constants relating to screen size.

The first routine (to fill the screen) just radiates lines of random colors radiating from the center of the screen to the edge, sweeping around the screen from the lower left hand corner. The circle radiating routine has been described above. The routine to radiate from the four corners just goes from corner to corner putting out randomly colored lines of random lengths at (you guessed it) random angles. Pressing any Ascii key will return to the menu, and a non-numeric input will abort the program with an IO-error. I should probably put some idiot-proofing code in at this point, but that's not what the program is demonstrating (and I'm too lazy to do it due to lack of interest).

The program **BEATLES** is an AUDIO demonstration, and does a Beatles medley including putting lyrics on the screen. Make sure that the CPU is running at a 2Mz max, otherwise the music will be unrecognizable, and all the notes will need to be changed.

The program RADIATE will require the large SYSTEM.LIBRARY since it USES Applestuff (Keypress, Random), RealModes and Transcend (Trig functions), and obviously PGRAF. BEATLES requires just APPLESTUFF.

```

0 REM ****
1 REM * Circling Demo -- by Dennis Cohen
2 REM *
3 REM * This program draws a single circle and rapidly
4 REM * changes colors of parts of the circle.
5 REM ****
10 ON ERR INVOKE"/BASIC/BGRAF.INV"
20 PERFORM initgrafix:OFF ERR

```

CROSSWORD-SCRAMBLER

...is a computer program that is educational and makes learning fun. Unlike many software products, **CROSSWORD-SCRAMBLER** lets you 'use your brain'. No, it's not a "shoot 'em up" type of arcade game...although you won't be disappointed by the graphic displays and musical interludes. Instead, if you like being human and would like to work with a computer (rather than SUBJECT yourself to one), then **CROSSWORD-SCRAMBLER** is what you have been waiting for.

Hundreds of different crossword questions will provide hours and hours of fun for the entire family. With **CROSSWORD-SCRAMBLER** you can turn your **Apple** /// into a true **Personal Computer** - one that is both powerful and entertaining.

CROSSWORD-SCRAMBLER is sold exclusively through **ON THREE** Magazine for only \$19.95.

```

30 PERFORM grafixmode(%1,%1)
40 PERFORM grafixon
45 PERFORM fillcolor(%0)
50 PERFORM fillport
120 maxloops%:=INT(RND(2)*5)+4
125 ON KBD GOTO 1000
130 FOR loopnum%:=1 TO maxloops%
150 FOR angle%:=0 TO 360 STEP 2
155 pc%:=16*RND(1):PERFORM pencolor(%pc%)
160 PERFORM moveto(%70,%96)
180 x%:=INT(30*COS(angle%))+70:y%:=INT(30*SIN(angle%))+96
190 PERFORM lineto(%x%,%y%)
200 NEXT angle%
210 NEXT loopnum%
220 TEXT:END
1000 IF KBD=27 THEN POP:TEXT:HOME:END
1010 ON KBD GOTO 1000
1020 RETURN

```

Continued on page 46.

THREE Tips by Bob Consorti

Do you have a short tip that you would like to share? Send them to **ON THREE** in care of this column so that others can benefit from the things everyone finds out about the ///.

Tip #1:

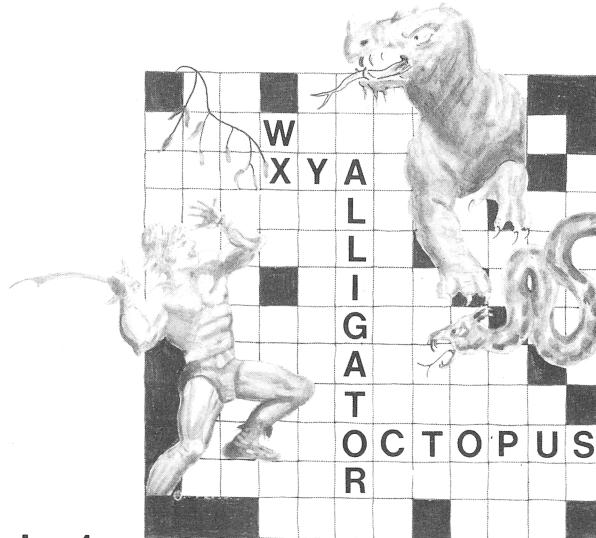
RPS information is available at dealers - NOT Apple Technical Support. (Sorry about that Steve - B.C.)

Tip #2:

Why don't the **Micro-Sci** drives work with CP/M version 2.21? Just a small bug. Change the device type of your Micro-Sci drivers from "E1" to "F1".

Tip #3:

Some **Micro-Sci A143's** were shipped with the most recent drivers but not the errata sheet that describes the changes. The latest version of the Micro-Sci Drivers is 1.3. Use them with **System Utilities V1.2**. Utilities V1.2 does not allow you to change the Device Subtype, therefore in version 1.3 of the Micro-Sci drivers they have added a one byte configuration block. Substitute the configuration block for the device subtype in the manual and make the changes to the device configuration block whenever it tells you to change the device subtype. ///



Lazarus /// (Undelete your deleted files!)

How much are your important data files worth? \$100, \$1000? Even if you back up your files regularly, the one file you accidentally delete will be the one you haven't ever backed up.

Wouldn't it be great if you could somehow regain those files you deleted? Well, with **Lazarus ///** - you can! Very easy to use, just insert the diskette with the files you deleted and **Lazarus ///** will recover it. A new standard in user friendliness, this program has on-line help and tutorial screens to aid in the use of the program. It even works with **ProFile** and other disk

ProFile ///
it in this
can't be
Use the

issue of
o get all
, all you

S!
ry and
re Disk
ce on all
no don't
lirectory
SOS and

amazing
hat your
th. Now
own on
her fonts
entation
is disk is
, all the
ny more

Board
issue of
fine the
ograms,
ing tool

If you would like to **Spread Sheet** from Basic or even use the program **Headfirst** to print heading for your program listings, this **Disk Of the Month** can do it for you!

DOM #4 - Emulation Patch

This disk contains all of the Pascal programs from the latest issue of **ON THREE**. Included are the **Apple II Emulation Patch** program, which lets you use any **Apple /// Font** in **Emulation Mode** and the new Pascal startup program for **Access ///** that lets you **AutoDial**!

Also on this **DOM** is the program and **UNIT** that lets you do

calculations from within your Pascal programs! Not forgetting the demos, we have also put the **Radiate Graphics Demo** and the **Beatles Music Demo** on this **DOM**. Even a number of pictures created by **Draw ON** are included. These pictures can be viewed with the program on **DOM #5**. It's also packed with programs and information that you can use!

DOM #5 - Access Draw ON!

This disk contains many of the utility programs in the latest issue of **ON THREE**. Included are the new Basic startup program for **Access ///** that lets you **AutoDial** and **Ben's SUPER Slot Machine**.

All of the **VisiCalc** and **WPL** programs are included as well as the **Circling Graphics Demo**. One of the nicest programs on this **DOM** is the **Draw ON /// Picture Demo**. It will show some of the beautiful pictures that can be created with **Draw ON**. Included on **DOM #4** and **DOM #5** are a dozen creations of **Draw ON** for your viewing pleasure.

For only \$9.95 (plus \$1.50 postage and handling) you can get any of these great packages. All five may be ordered for the extra low price of just \$37.50 (plus \$2.00 for postage and handling). Mix and match any of the **DOM**'s for the low prices shown below. Order today!

Bulk and group purchase rates are as follows (any mix of **DOM**'s):

2-9 disks: \$7.50 apiece + \$2 total shipping

10-24 disks: \$7.00 apiece + \$3 total shipping

over 24 disks: \$6.50 apiece + \$4 total shipping

ON THREE O'Clock

Calling all you time conscious **Apple ///** owners out there? How would you like a working **clock/calendar** for your **Apple ///**? Just as it was originally intended (and now included in the **Apple /// plus**), this kit comes with a plug in clock chip with a battery backup.

With an **ON THREE O'Clock** installed, any time you save or modify a file, the current time and date will be stored on disk. Thus you will now be able to tell which file you last worked on. Your programs can now use the **Apple ///** built-in date and time routines to give you an up to the second read-out of what time it is.

Extremely easy to install and adjust, it is completely compatible with **SOS** and doesn't use up a slot! This is the one you have been waiting for! The package contains comprehensive instructions and a **Six Month Warranty**! Try and get that deal anywhere else!

Coupled with the **ONTIME** clock driver (see the ad on inside front cover), the **ON THREE O'Clock** is an invaluable tool for everyone who has an **Apple ///**. What's the best part? - The price! While others are selling their clock for \$60 and up, we have broken the \$50 barrier. Heck, we broke the \$40 barrier!

For only \$39.95 (plus \$2.50 for postage and handling) you can get the best little clock in town! See the **Special Offer** on the inside front cover for the **ONTIME - ON THREE O'Clock**

2-9 clock sets: \$36.50 apiece + \$5 total shipping

10-24 clock sets: \$33.25 apiece + \$7 total shipping

over 24 clock sets: \$31.00 apiece + \$9 total shipping

Bulk and group purchases must have one mailing address. Please use the attached envelope for orders. If the envelope is missing, send to:

ON THREE
Attn: ORDER DEPT.
P.O. Box 3825
Ventura, California 93006

Or call (805) 644-3514 direct for a Visa or MasterCard order.

Three Shorts

(Well, they're not that long!)

by Dennis Cohen

There are three programs listed here, two in **Pascal** and one in **BASIC**. The BASIC program, **CIRCLE**, requires that **/BASIC/BGRAF.INV** be online and is really just a part of the Pascal program **RADIATE**, and generates randomly sized and located circles of multiple, constantly changing colors on the screen.

RADIATE is a menu-driven collection of three graphics demos taken from Tom Swan's UCSD Turtlegraphics-based Moire routines and modified to take advantage of some of the ///'s rather unique capabilities. All three of the demos use the COL140 mode, so that if you want to use CP280 you'll need to change some global constants relating to screen size.

The first routine (to fill the screen) just radiates lines of random colors radiating from the center of the screen to the edge, sweeping around the screen from the lower left hand corner. The circle radiating routine has been described above. The routine to radiate from the four corners just goes from corner to corner putting out randomly colored lines of random lengths at (you guessed it) random angles. Pressing any Ascii key will return to the menu, and a non-numeric input will abort the program with an IO-error. I should probably put some idiot-proofing code in at this point, but that's not what the program is demonstrating (and I'm too lazy to do it due to lack of interest).

The program **BEATLES** is an AUDIO demonstration, and does a Beatles medley including putting lyrics on the screen. Make sure that the CPU is running at a 2Mz max, otherwise the music will be unrecognizable, and all the notes will need to be changed.

The program RADIATE will require the large SYSTEM.LIBRARY since it USES Applestuff (Keypress, Random), RealModes and Transcend (Trig functions), and obviously PGRAF. BEATLES requires just APPLESTUFF.

```

0  REM ****
1  REM * Circling Demo -- by Dennis Cohen
2  REM *
3  REM * This program draws a single circle and rapidly
4  REM * changes colors of parts of the circle.
5  REM ****
10 ON ERR INVOKE"/BASIC/BGRAF.INV"
20 PERFORM initgrafix:OFF ERR

```

CROSSWORD-SCRAMBLER

...is a computer program that is educational and makes learning fun. Unlike many software products, **CROSSWORD-SCRAMBLER** lets you 'use your brain'. No, it's not a "shoot 'em up" type of arcade game...although you won't be disappointed by the graphic displays and musical interludes. Instead, if you like being human and would like to work with a computer (rather than SUBJECT yourself to one), then **CROSSWORD-SCRAMBLER** is what you have been waiting for.

Hundreds of different crossword questions will provide hours and hours of fun for the entire family. With **CROSSWORD-SCRAMBLER** you can turn your **Apple** /// into a true **Personal Computer** - one that is both powerful and entertaining.

CROSSWORD-SCRAMBLER is sold exclusively through **ON THREE** Magazine for only \$19.95.

```

30 PERFORM grfxmode(%1,%1)
40 PERFORM grfxon
45 PERFORM fillcolor(%0)
50 F
120
125
130
150
155
160
180
190
200
210
220
1000
1010
1020

```

TH

Do you
can be
Tip #1

RPS in
Support

Tip #2
Why
2.21? Ju
drivers f

Tip #3
Some
drivers t
latest ve

System
change

Sci drive
Substitu

manual
whenev

ON THREE
ATTN: ORDER DEPT.
P.O. BOX 3825
VENTURA, CA 93006

BUSINESS REPLY MAIL
FIRST CLASS
PERMIT NO. 90
VENTURA, CA
POSTAGE WILL BE PAID BY ADDRESSEE

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



Lazarus /// (Undelete your deleted files!)

How much are your important data files worth? \$100, \$1000? Even if you back up your files regularly, the one file you accidentally delete will be the one you haven't ever backed up.

Wouldn't it be great if you could somehow regain those files you deleted? Well, with **Lazarus ///** - you can! Very easy to use, just insert the diskette with the files you deleted and **Lazarus ///** will recover it. A new standard in user friendliness, this program has on-line help and tutorial screens to aid in the use of the program. It even works with **ProFile** and other disk drives!

This program gives a first look at a new type of **Apple ///** program - **incredibly easy** to use. Described in detail in this issue, **Lazarus ///** is one of the few programs that you can't be without. The best part? - **Lazarus /// is only \$29.95**. Use the enclosed order form or call and place your order today!

Disk Of the Month

Do you have the time to type in the programs in each issue of **ON THREE**? Wouldn't it be great if there was a way to get all the programs without having to type them in? - There is, all you have to do is order the disk!

DOM #1 - Extra Disk Space Plus!

This disk contains all the programs in the **January and February-March** issues of **ON THREE**. Included are **Disk Pak1**, which will give you four extra blocks of disk space on all your data disks (a very handy feature for those of you who don't have a hard disk!); **Disk Pak2**, which lists the files on a directory using Pascal; all of the **Graphics and Sound Demos** and much, much more!

DOM #2 - Changing the Characters Of Your Printer

This disk contains a program that will do a most amazing thing, it will enable you to change the characters that your **Apple Dot Matrix** (or **ProWriter**) printer prints with. Now your **DMP** can print with the same characters that are shown on your text screen. **Fancy Gothic letters** and many other fonts are available to use on your printer. Complete documentation makes this program very easy to use. Also included on this disk is a program to list the files on your **Apple II** diskettes, all the other programs from the **April-May** issue, and many more Graphic demonstrations.

DOM #3 - Changing Your Keyboard

This disk contains all the programs in the **June-July** issue of **ON THREE**. Included is the program that lets you redefine the positions of the keys on your keyboard, all of the **WPL** programs, the **Disk Formatting Utility**, the **Graphics Sketching** tool and everything else!

If you would like to **Spread Sheet** from Basic or even use the program **Headfirst** to print heading for your program listings, this **Disk Of the Month** can do it for you!

DOM #4 - Emulation Patch

This disk contains all of the Pascal programs from the latest issue of **ON THREE**. Included are the **Apple II Emulation Patch** program, which lets you use any **Apple /// Font** in **Emulation Mode** and the new Pascal startup program for **Access ///** that lets you **AutoDial**!

Also on this **DOM** is the program and **UNIT** that lets you do

calculations from within your Pascal programs! Not forgetting the demos, we have also put the **Radiate Graphics Demo** and the **Beatles Music Demo** on this **DOM**. Even a number of pictures created by **Draw ON** are included. These pictures can be viewed with the program on **DOM #5**. It's also packed with programs and information that you can use!

DOM #5 - Access Draw ON!

This disk contains many of the utility programs in the latest issue of **ON THREE**. Included are the new Basic startup program for **Access ///** that lets you **AutoDial** and **Ben's SUPER Slot Machine**.

All of the **VisiCalc** and **WPL** programs are included as well as the **Circling Graphics Demo**. One of the nicest programs on this **DOM** is the **Draw ON /// Picture Demo**. It will show some of the beautiful pictures that can be created with **Draw ON**. Included on **DOM #4** and **DOM #5** are a dozen creations of **Draw ON** for your viewing pleasure.

For only \$9.95 (plus \$1.50 postage and handling) you can get any of these great packages. All five may be ordered for the extra low price of just \$37.50 (plus \$2.00 for postage and handling). Mix and match any of the **DOM**'s for the low prices shown below. Order today!

Bulk and group purchase rates are as follows (any mix of **DOM**'s):

2-9 disks: \$7.50 apiece + \$2 total shipping

10-24 disks: \$7.00 apiece + \$3 total shipping

over 24 disks: \$6.50 apiece + \$4 total shipping

ON THREE O'Clock

Calling all you time conscious **Apple ///** owners out there? How would you like a working **clock/calendar** for your **Apple ///**? Just as it was originally intended (and now included in the **Apple /// plus**), this kit comes with a plug in clock chip with a battery backup.

With an **ON THREE O'Clock** installed, any time you save or modify a file, the current time and date will be stored on disk. Thus you will now be able to tell which file you last worked on. Your programs can now use the **Apple ///** built-in date and time routines to give you an up to the second read-out of what time it is.

Extremely easy to install and adjust, it is completely compatible with **SOS** and doesn't use up a slot! This is the one you have been waiting for! The package contains comprehensive instructions and a **Six Month Warranty**! Try and get that deal anywhere else!

Coupled with the **ONTIME** clock driver (see the ad on inside front cover), the **ON THREE O'Clock** is an invaluable tool for everyone who has an **Apple ///**. What's the best part? - The price! While others are selling their clock for \$60 and up, we have broken the \$50 barrier. Heck, we broke the \$40 barrier!

For only \$39.95 (plus \$2.50 for postage and handling) you can get the best little clock in town! See the **Special Offer** on the inside front cover for the **ONTIME - ON THREE O'Clock**

2-9 clock sets: \$36.50 apiece + \$5 total shipping

10-24 clock sets: \$33.25 apiece + \$7 total shipping

over 24 clock sets: \$31.00 apiece + \$9 total shipping

Bulk and group purchases must have one mailing address. Please use the attached envelope for orders. If the envelope is missing, send to:

ON THREE

Attn: ORDER DEPT.

P.O. Box 3825

Ventura, California 93006

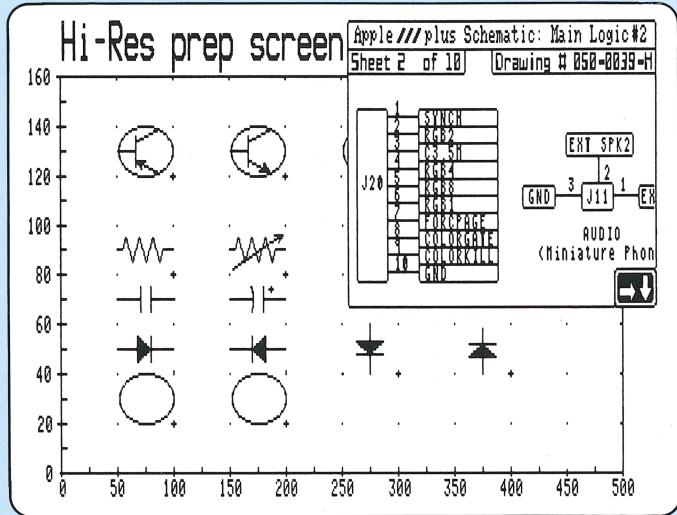
Or call (805) 644-3514 direct for a Visa or MasterCard order.

ON THREE

Proudly announces...

Draw ON /// is a powerful and versatile graphics tool designed exclusively for the Apple /// and Apple /// plus computers. **Draw ON ///** transforms your Apple /// into a combination drafting table, easel and sketch pad. **Draw ON** works in all of the Apple ///'s color and Black/White graphics modes and brings the power of **LisaDraw** and **MacPaint** to your ///.

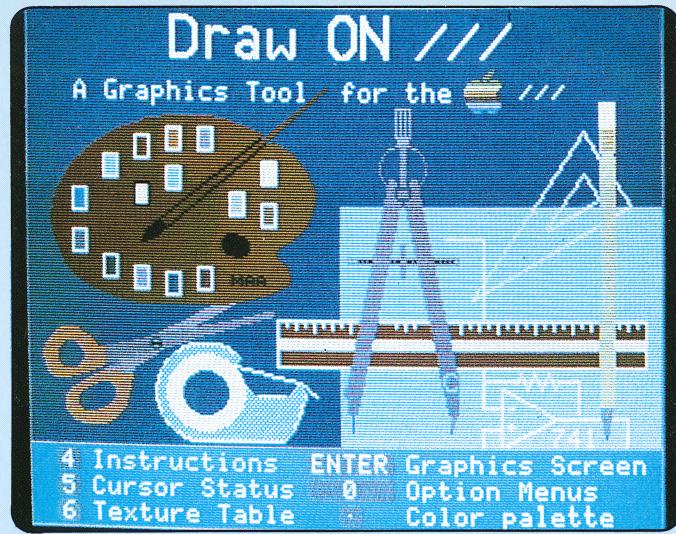
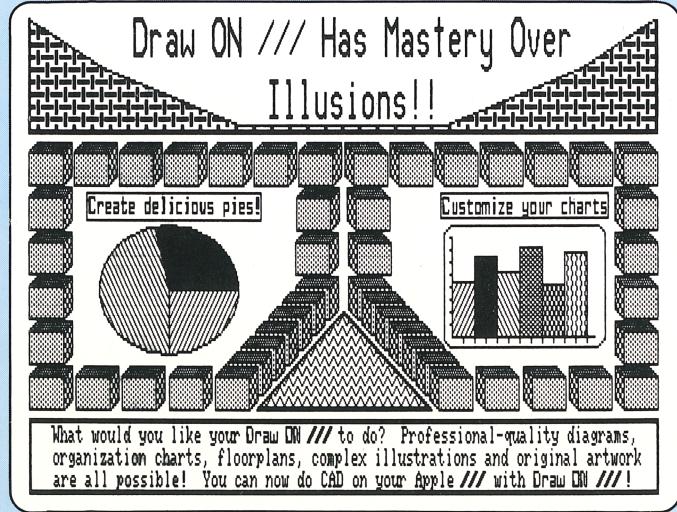
Draw ON /// provides powerful cut and paste facilities with the ability to create unlimited libraries of your own special figures and objects (such as circuit components, business logos, animation characters...). You may select these objects and move them onto, between, and around any of the many drawing screens.



Objects may be "shrunk" or expanded, rotated, textured and inverted. They may be moved on top of objects already in the screen, behind previously drawn objects, or overlaid along with other figures on the screen. You can also zoom in on a particular portion of the screen to do detailed work.

The screen that **Draw ON** uses is a window into a much larger document. **Draw ON** allows you to scroll to different parts of the document at will. With the addition of the popular **PKASO** Printer Interface, **Draw ON** can directly print your drawings to a variety of Dot-Matrix printers. The Black and White Pictures on this page were each designed in less than 30 minutes and printed on a standard Apple Dot-Matrix printer. (To print the color pictures you will need an appropriate color printer such as the **IDS Prism**.)

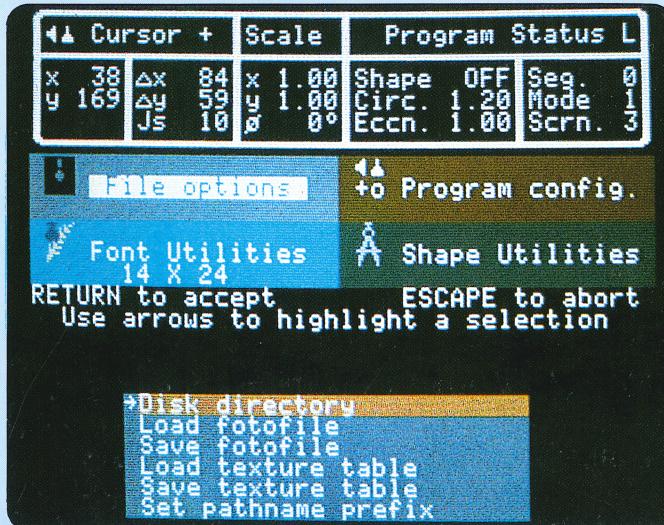
On the color drawing screens you may paint with the Apple ///'s 16 colors and over 1000 pseudo color-texture combinations. And on all of the drawing screens there is no limit to the types of "Brush" that you can use. Simply pick-up any object and draw with it.



You can also mix text directly with your drawings. A variety of fonts are supplied in sizes up to 14 x 24. If you don't like any of the typefaces that come with **Draw ON** you can design your own! You can label your drawings with any of these fonts and even use them in your other programs.

Features such as "rubber-banding" of lines, user adjustable grids, built-in help screens and easy to follow menus make **Draw ON ///** the ONLY graphics package for the Apple /// that is both powerful and simple to use. Combined with an excellent instruction manual, you can be doing useful work in less than an hour. The only limit to what you can do with **Draw ON** is your imagination.

Draw ON gives an individual the power of a graphic arts studio. Use it in creating charts, preparation of slides and tables for presentations, and letterhead design. With **Draw ON** you can make changes to the dull graphs that your other programs create by adding borders, textures and different typefaces. Even Computer Aided Design applications such as circuit layouts, drafting, and flowcharting are now possible on your Apple /// with **Draw ON ///**.



Program Requirements:

Apple /// with 256K or an Apple /// plus
B/W Monitor (A Monitor /// is recommended)

Optional Equipment:

Cursor /// Joystick (or equivalent)
RGB Color Video Monitor
PKASO Printer Interface
Dot-Matrix Printer

Suggested retail price: \$129

Available at finer computer stores everywhere
and directly from **ON THREE**.

Dealer inquiries invited.

Available Now!